DESIGN AND IMPLEMENTATION OF AN ENERGY-EFFICIENT FULL-COLOR LARGE-AREA LED DISPLAY SYSTEM

LV XUECONG

M.Phil

The Hong Kong Polytechnic University

2015

The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

DESIGN AND IMPLEMENTATION OF AN ENERGY-EFFICIENT FULL-COLOR LARGE-AREA LED DISPLAY SYSTEM

LV XUECONG

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Philosophy

August 2014

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____________________(Signed)

____LV Xuecong______(Name of Student)

To my family

Abstract

Red-green-blue (RGB) light-emitting-diode (LED) is more capable to vary its color in real time, less energy-consuming and more durable than the conventional light source, and thereby has gained increasing popularity in various applications. One of the promising applications of RGB LED is the full-color large-area LED display system. Large-scale LED video displays market has competitive advantages, since few technologies are capable to deliver large size at high contrast and durability. Due to the higher power consumption of the full-color large-area LED display, maximizing the power efficiency has become a major design objective of this type of display system.

This thesis demonstrates how the multi-level pulse-width modulation (MPWM) driving method can be implemented on a full-color large-area LED display system through a low-cost modification of the existing hardware configuration. This method uses multiple LED driver modules that are linked in parallel to transfer a MPWM current to individual LED pixels. For a three-level driving scheme, the criterion for selecting the mid-level current in order to maximize the luminous efficacy gain is derived mathematically. The algorithms of video signal conversion, which are designed to derive the sub-video signals from the original high-definition multimedia interface (HDMI) or digital visual interface (DVI) video signals for activating multiple LED driver modules, are also discussed.

Publications

Journal paper

X. Lv, K. H. Loo, Y. M. Lai, Chi K. Tse, "Energy-Saving Driver Design for Full-Color Large-Area LED Display Panel Systems," *IEEE Transactions on Industrial Electronics*, Vol. 61, No. 9, pp. 4665–4673, 2014

Conference paper

X. Lv, K. H. Loo, Y. M. Lai, Chi K. Tse, "Energy Efficient LED Driving System for Large-Scale Video Display Panel," *39th Annual Conference of the IEEE Industrial Electronics Society*, IECON 2013, Vienna, Austria, 11–13 November 2013

Acknowledgement

It is my honor to pursue my MPhil study at the Hong Kong Polytechnic University. During the past three years, I have gained research skills that form the foundation of my research career, obtained valuable experience in tackling difficulties, and gained sincere friendships that always support me.

First, I would like to express my deepest gratitude to Dr. Y.M. Lai, Dr. K.H. Loo and Dr. S.K. Ip, my supervisors for their sustained encouragement, their patient guidance, and valuable advice throughout the course of my study. During the past three years, their support motivated me when I made progress, and their encouragement fuelled me when I met difficult situations. Without their professional instructions, illuminating insights, and constructive suggestions, this thesis could not have been completed. Besides their extensive knowledge, I personally benefit more from their rigorous scholarship, kindness, optimism and humor. All in all, I have been incredibly fortunate to have Dr. Y.M. Lai, Dr. K.H. Loo and Dr. S.K. Ip as my supervisors!

Second, I would like to vote my thanks to Prof. C. K. Michael Tse and the members of the Applied Nonlinear Circuits and Systems Research Group for their kind support. I am also thankful for my lab mates for their assistance, and they include Ka Hei Leung, Chi Shing Wong, Ling Ling Cao, Fei Luo, Joseph Pang and Jamin Tse. When I feel lost in life or study, they are always my good friends who encourage me and help me. I really cherish the joyful time with them.

Last but not least, I am thankful for my parents and Junyi. Without their support and love, I would not be able to accomplish so much in my MPhil study. Thanks for their great confidence in me all through these years and for companying me to go through the whole process.

Table of Contents

Abstract..4

Publications..5

Acknowledgement 6

List of figures 10

List of tables...13

Chapter 1 Introduction..14

1.1 Motivation..14

1.2 Literature review..22

1.2.1 Overview of video display 22

1.2.2 Dimming methods for LED video display 28

1.3 Objective..38

1.4 Outline of the thesis 39

Chapter 2 LED video display and its dimming methods............................40

2.1 Basics of LED video display 40

2.2 System architecture..44

2.3 The proposed multi-level dimming method.......................................50

2.4 The algorithms of video data correction 56

2.4.1 The correction algorithm for MPWM 56

2.4.2 The anti-gamma correction algorithm..58

2.5 Summary..63

Chapter 3 LED video display design and implementation.........................65

3.1 Modified system architecture...65

3.2 Hardware implementation...70

3.3 Software implementation...78

3.3.1 Architecture of FPGA controller 78

3.3.2 Simulation results 81

3.4 Experimental verification 84

3.4.1 Experimental results on energy saving methods 85

3.4.2 Verification of LED display system performance 91

3.5 Realization of large-area LED display system 100

3.6 Summary 103

Chapter 4 Conclusion 105

4.1 Contribution of this thesis 105

4.2 Suggestions for future work 108

Appendix A 110

Appendix B 112

References 137

List of figures

Figure 1 The basic principle of LED driving technique. 17

Figure 2 Examples of LED application [20]. 19

Figure 3 An example of LED backlight LCD display [26]. 20

Figure 4 The block diagram of a LED display system [45]. 28

Figure 5 Conventional full-color video data dimming method. 32

Figure 6 The explanation of bitwise separate method. 33

Figure 7 The explanation of linear PWM operation method. 34

Figure 8 The comparison between PWM and BPWM with the same weight values. 34

Figure 9 The comparison between BPWM and GPWM with the same weight values. 36

Figure 10 Architecture of basic LED video display system. 42

Figure 11 The smallest unit of LED display module which consists of RGB LEDs. 43

Figure 12 An example of a full-color LED video display panel shown outside the gym [20]. 44

Figure 13 An FPGA-controlled LED display panel system architecture. ... 45

Figure 14 An example of timing diagram showing the various output signals generated by video decoder. 46

Figure 15 Separation of data bits according to their weights (bit-plane separation). 48

Figure 16 A typical LED driver module. 49

Figure 17 An example of timing diagram showing the various control signals applied to the LED driver module. 50

Figure 18 Dimming curves associated with three driving methods. 51

Figure 19 Typical current waveforms for three driving methods. 52
Figure 20 Maximization of area A leads to the maximum gain in luminous efficacy for MPWM driving over conventional PWM driving. 54
Figure 21 PWM to MPWM current conversion. 57
Figure 22 The relationship between the normalized control signal and normalized illumination in CRT display. 60
Figure 23 The relationship between the normalized control signal and normalized illumination in LED display. 61
Figure 24 Anti-gamma correction. 63
Figure 25 Block diagram showing the modified system architecture. 66
Figure 26 Implementation of three-level MPWM using two LED driver modules connected in parallel. 67
Figure 27 Architecture of video source decoding design. 71
Figure 28 Circuit diagram of conventional LED video display module. 74
Figure 29 Circuit diagram of modified LED video display module. 74
Figure 30 Resistance of the external resistor. 75
Figure 31 Interception of video source. 76
Figure 32 The front view and back view of the a single LED module consisting of 32 × 32 LED pixels, Fig. 32(a) represents the front view, and Fig. 32(b) represents the back view. 77
Figure 33 Energy-efficient full-color LED display system architecture. 77
Figure 34 The architecture of FPGA controller system. 79
Figure 35 Structure of HDMI timing module. 81
Figure 36 The simulated structure of HDMI interface module. 82
Figure 37 Simulated results of bit-plane conversion. 82
Figure 38 Details of simulated results of bit-plane. 83
Figure 39 Simulated results of interfaces of LED display. 83

Figure 40 Details of simulated results of interfaces of LED display........... 84

Figure 41 Average luminance of the (a) red, (b) green, and (c) blue LED subpixels.. 87

Figure 42 The luminance output of red LED. 88

Figure 43 The luminance output of green LED. 88

Figure 44 The luminance output of blue LED. 89

Figure 45 For each of the three colors, dashed lines indicate the maximum achievable luminous efficacy gain (by using dc driving) over PWM driving, and solid lines with marks indicate the luminous efficacy gain achieved by using MPWM driving under various junction temperatures. 90

Figure 46 Setup of experiment for RGB LEDs. 90

Figure 47 Prototype of LED display panel consisting of 2 ×3 LED display modules or 64 ×96 LED pixels. 91

Figure 48 The luminance uniformity test with solid colors. 93

Figure 49 The luminance differences between each of the modules for the color as shown in Fig.45(a). 95

Figure 50 The luminance differences between each of the modules for the color as shown in Fig.45(b). 95

Figure 51 The luminance differences between each of the modules for the color as shown in Fig.45(c). 96

Figure 52 LED video display with good performance 97

Figure 53 LCD and LED video display of flags in real-time. 98

Figure 54 Setup of MPWM based LED video display system.................... 99

Figure 55 Large-scale LED display system consisting of standard LED display panels. 101

Figure 56 The software realization of large-scale LED display system.... 103

List of tables

Table 1 The comparison of incandescent lamps, Halogen lamps, CFLs, and LEDs [15]. 16

Table 2 Comparison of Cyclone III FPGA family. 72

Table 3 Specifications of the LED video display module circuit. 75

Table 4 The controller tools. 84

Table 5 Specification of the full-color LED video display system. 99

Table 6 Description of the input and output of FPGA controller system. ..110

Chapter 1 Introduction

1.1 Motivation

Electric light source has been continuously developing with substantial improvements since the first electric light source, incandescent light bulb, was invented by Swan and Edison in 1879 [1, 2]. The incandescent light bulb is the first artificial light source that used a filament heated by the electric current passing through it to produce light. However, the filament suffered from having a short lifespan and therefore was replaced by other materials. In order to increase the lifespan of the filament, the halogen lamp is produced which is filled with a small amount of halogen gas, such as chlorine, iodine or bromine. A halogen lamp can be operated at a higher temperature and has a longer lifespan than the ordinary incandescent light bulb [3]. However, both the incandescent light bulb and the halogen lamp have low efficiency, as they convert only 5% of the energy they produce into light and 95% into heat [4].

To improve the energy efficiency for general lighting, the fluorescent tubes were introduced by the General Electric Company in 1930s [5]. The fluorescent tubes can convert more than 30% of energy they produce into visible light, so they have higher energy efficiency than the incandescent bulbs, and thereby the fluorescent tubes were widely used in European countries, the U.S.A and some other countries to replace incandescent bulbs [6]. Unfortunately, the fluorescent tubes also have an obvious drawback that they cannot be connected to dimmer switches designed for the incandescent bulbs. This drawback results from the fact that the interaction between the waveforms of the voltage generated by the dimmer and the ballasts is ineffective and the fact that the arc in the fluorescent tube has difficulty in maintaining stable at a low power level. The compact fluorescent lamp (CFL) was later proposed to avoid the drawback of traditional fluorescent tubes, as CFL could work properly under the condition of low power supply [7]. In general, both the fluorescent tube and the CFL are categorized as low-pressure gas-discharge lamps, and their energy efficiencies remain at low levels without any breakthrough at that time. The high-intensity discharge (HID) lamp with high-pressure gas was later proposed to obtain higher energy efficiency than that of the CFL [8]. However, HID lamps still have some disadvantages. One disadvantage is that they are fragile since their filaments are easily burnt out. Another disadvantage is that they produce faint light when first kindled, and require a few seconds (e.g., 15 - 20 seconds) to achieve the full intensity of light [9].

Unlike conventional light sources, solid-state lighting (SSL) employs the semiconductor light-emitting diode (LED) as source of illumination. The first solid-state electroluminescence was discovered by H.J. Round in the Marconi Lab, and its light was delivered from a silicon carbide (SiC) crystal when a large voltage was applied [10]. In 1962, the first red-color LED based on the gallium arsenide phosphide (GaAsP) alloy was discovered by N. Holonyak in the General Electric Company. However, the first-generation LED had very low efficiency in light output, so it was merely used as the indicator light. Due to the development of materials and architectures of LED, the aluminum gallium indium phosphide (AllnGaP/GaP) based red LED , having more than ten times efficiency compared with that of the traditional incandescent lamps, was introduced in 1990 [11]. In 1993, the first blue-color LED based on the indium gallium nitride (InGaN) alloy was discovered by Nakamura [12, 13]. In the following year, the green-color LED was also invented by Nakamura [14]. In 1996, the first white LED was developed by Nakamura and Nichia, and it was based on the blue-color LED coated with yellow phosphors on the top layer of the LED. A comparison among incandescent lamps, Halogen lamps, CFL, and LED is illustrated in Table 1.

Table 1 The comparison of incandescent lamps, Halogen lamps, CFLs, and LEDs [15].

Comparison among incandescent lamps, Halogen lamps, CFL, and LED				
	Incandescent	Halogen	CFL	LED
Purchase price	$0.41	$1.50	$0.99	$6.97
Power used (watts)	60	43	14	9.5
Lifespan (hours)	1,000	2,500	10,000	25,000
Lumens (mean)	860	750	775	806
Color temperature	2700	2900	2700	2700

(Kelvin)				
Total cost per 860 lumens	$360	$356	$94	$73
Lumens/watt	14.3	17.4	55.4	84.2

LEDs are p-n junction devices consisting of gallium phosphide (GaP) and gallium nitride (GaN). The characteristics of LEDs are similar to the diodes. The LEDs can become visible photons when the electrons are driven into the active region by a fitting voltage, and then release energy. Since LEDs are current-driven devices, the forward current flowing through the devices is utilized to control the luminance intensity. Like the normal diodes, the LEDs process the current in one direction. So, the simplest way to dim the LEDs can be achieved by using the continuous direct current (DC) which corresponds to the conventional amplitude mode dimming technique, as illustrated in Fig. 1.

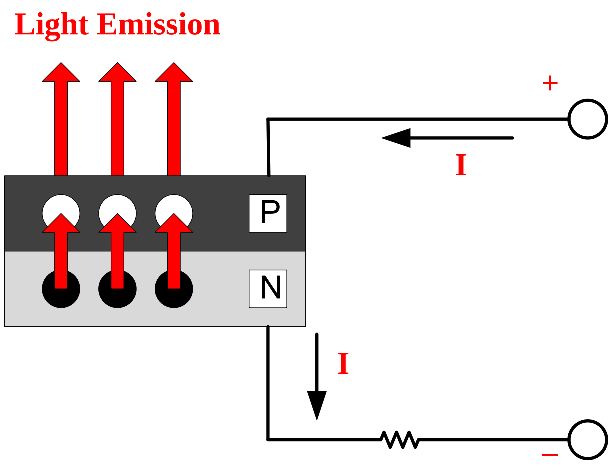


Figure 1 The basic principle of LED driving technique.

Compared with the conventional lighting sources, LEDs consume less power. For the conventional lighting sources, the majority of the energy generated is lost in the form of heat, and little remaining energy is converted into visible lighting [16]. In LEDs, recombination of injected holes and electrons take place in the semiconductor junction. This design has the advantage that a large proportion of the energy the LEDs produce is converted into visible light [16]. However, the single LED die can only emit monochromatic light, so more LED dies need to be combined together to achieve more colors. For instance, red-green-blue (RGB) LEDs contain three different LED emitters. Three LEDs (e.g., red, green and blue) are often packaged in one RGB LED which has four wires----a common lead (anode or cathode) and others are for red, green and blue. One advantage of the RGB LEDs is that they can provide a wide range of colors and have more flexibility with respect to the color adjustability for lighting applications [17]. In addition, the RGB LEDs have the advantages of high brightness and saturated chromaticity, and they are generally more high-efficiency, fast-switching and long-lifespan than conventional lightings sources.

The LEDs have been widely used in various applications, for example, single-color LEDs have been employed in poles, parking garages and other architectural lighting [18]. Most of the single color LEDs are used as indicators which contain LEDs of a variety of sizes ranging from 2 mm to 10 mm and using through-hole or surface-mount technologies. Single-color LEDs are appropriate for applications subject to constant on-off cycling. They are superior to fluorescent lamps and HID lamps, since they can sustain frequent on-off cycling and only require a short time before restarting. The low energy consumption, the economical maintenance cost and the relatively small size of LEDs make it possible to use LEDs as status indicators and to display them on a variety of equipment and installations. Single-color LEDs are also well suited for further applications in information transmission as traffic lights and signals, exit signs, emergency vehicle lightings and ships' navigation lights [19]. For instance, the application of single-color LEDs in brake may improve traffic safety, since the time needed to light fully for single color LEDs is about 0.5 second shorter than that needed to light fully for an incandescent bulb, providing drivers behind more time to respond to the brake. Fig. 2 presents some typical applications of LEDs.

Figure 2 Examples of LED application [20].

Besides single-color LEDs, RGB LEDs begin to be employed in a new range of applications. Since RGB LEDs can provide a full visible color spectrum and have the characteristics of high brightness, low voltage power supply and high dimming ratio, they can be applied for backlight of LCD display panels, lightweight laptop displays and light sources for projectors [21, 22]. Screens for the television and computer monitors can be manufactured thinner using RGB LEDs as the backlight [23, 24]. RGB LEDs are more energy-efficient than the cold cathode fluorescent lamp (CCFL) as backlights. For instance, a 15-inch computer notebook's backlight using CCFL consumes 5 W electricity in an hour, while the backlight using RGB LEDs consumes less than 3 W within the same amount of time. Another important usage of RGB LEDs is to provide the local dimming technique for LCD video display. The local luminance intensity of the LED-backlight LCD video display can be linked to the image displayed, enhancing the contrast ratio of the display [25]. At the same time, the LCD's power consumption can be further reduced under 2 W, representing more than 60% improvement than conventional one. Fig. 3 shows a typical LED-backlight LCD video display architecture. This kind of LCD video display is not self-illuminating (i.e. it requires a source of light for illumination). When RGB LED backlighting is used, it can produce a dynamic illumination for the video display. LED-backlight LCD video display can produce deep dark when LEDs are turned off and can produce high brightness when LEDs are turned on.

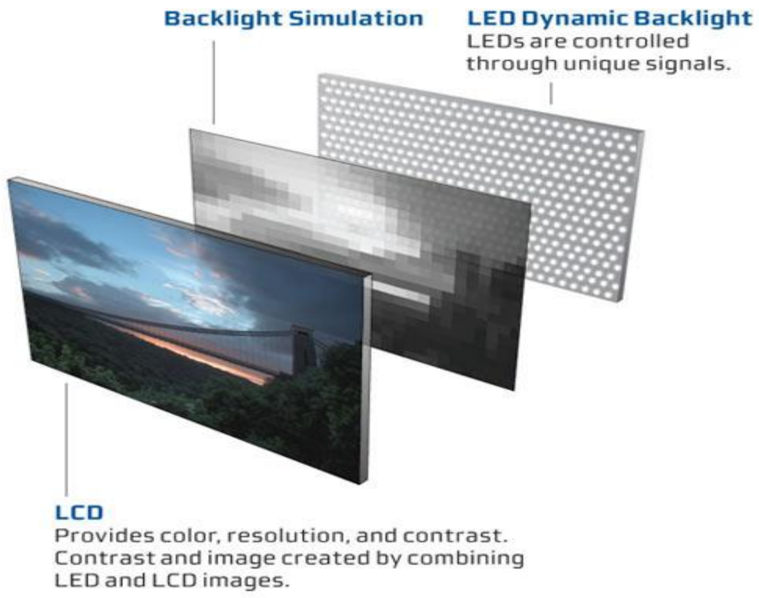

Backlight Simulation
LED Dynamic Backlight
LEDs are controlled through unique signals.
LCD
Provides color, resolution, and contrast. Contrast and image created by combining LED and LCD images.

Figure 3 An example of LED backlight LCD display [26].

With the increasing industrial development, there is a rising demand for energy consumption in many countries. Governments and enterprises are willing to seek for new energy sources and effective ways to utilize energy. Based on the data from the U.S. department of energy, approximately 30% electrical power has been consumed by commercial and industrial lighting [4]. Meanwhile, according to the world energy organization, over 70% of the energy consumed by lamps and displays is converted into heat. Reducing the power dissipation on the existing general illumination light source is an effective way for energy saving. Therefore, studies on energy saving on lighting deserve more attention from academic researchers.

LED video displays can be the future of the large-area display market. Their scale, durability, and ability to produce eye-catching video make them integral to the experience in stadiums, arenas, entertainment complexes, and outdoor advertising across a wide range of uses and industries [28]. LED displays are versatile and can be used to excite customers, enhance fan experience, raise company awareness and promote brands. Due to the high power consumption of large-area LED arrays for full-color video displays, maximizing the power efficiency has become a major design objective of this type of display system. In particular, the luminous efficacy of the LED arrays should be improved as they are the main consumer of electrical power in the entire system, the power consumption of the video processing circuitry is low, and the onboard switched mode power supplies can be designed with very high efficiencies.

Moreover, although there are numerous studies reporting the influence of forward current on the efficacy of RGB LEDs, the optical characteristics of RGB LEDs on LED video display system still remain unknown and the practical uses of RGB LEDs dimming techniques need further exploration

[29]. So, in this thesis, the display performance of RGB LEDs dimming under AM technique and PWM technique will be investigated experimentally and the relation between efficiency and the driving technique will be explained. The limited dimming range of bi-level PWM dimming technique and the complex circuit implementation of *n*-level PWM dimming technique are main challenges for RGB LEDs display technology. Therefore, this study intends to investigate the design and implementation of full-color large-area LED display system in an energy-efficient manner by improving the dimming methods for LED displays.

1.2 Literature review

In this section, an overview of video display and a detailed review of previous studies on dimming methods are given.

1.2.1 Overview of video display

Since the cathode ray tube (CRT) is cost-efficient, its application has been proliferated in video display systems applied in the enterprise information systems, the desktop publishing system and the office-automation system [30]. The CRT consists of a electron emitter and a screen. The electron beams generated by the electron emitter are accelerated and deflected onto the screen to display images. The horizontal synchronization (HS) signals manipulate the movement of electron beams from left to right of the screen, whereas the vertical synchronization (VS) signals control the transfer of electron beams from the top to bottom. The CRT display presents the video images via the raster scan. The raster scan consists of progressive scan and interlaced scanning [31]. In the progressive scan, the electron beams firstly reach the top left corner of the screen and then moves to the top right corner of the screen point by point. Next, the electron beams moves to the next horizontal line and reaches the screen from left to right as well. This process repeats until the electron beams reaches the bottom right of the screen. So the first field can be presented on the screen, and the electron beams can start scanning the next field from the top left of the screen. On the other hand, in the interlaced scanning, the electronic beams scan every other line firstly, and then scan the rest of the lines. The interlaced scan may make the screen flicker more frequently than the progressive scan, leading to visual fatigue. So most CRT display uses the progressive scan.

The image information processed by the computer graphics card can be transmitted to screen via the graphics output interface [32], which is a connection between the computer and the screen, and responsible for transmission of image signals to the display screen. Due to its unique design and manufacture, the CRT monitor can only process analog signals. The video graphics array (VGA) interface can process the analog signals output by the graphics card [33]. These signals are transmitted through the VGA cable to the display device. They contain the image display information digitally generated inside the computer, and can be converted to RGB signals by the digital/analog (D/A) converter [34]. For analog CRT monitors, signals are directly transmitted to the corresponding signal processing circuit, a drive tube to generate an image.

However, the CRT devices have several drawbacks. Gorog [35] reported that the CRT monitor often emits much during the operation, and thereby has relatively high power consumption. Chuang et al. [36] demonstrated that the maximum size for direct view displays (e.g., CRT display) is limited to about 40 inches because of practical and manufacturing restriction. Lu et al. [37] showed that even though the size of the CRT device can be expanded through an array of separate displays, CRT display with a large size may weigh about 300 pounds, thus suffering from such problems as relocation and resettlement. The CRT devices are subject to magnetic interference, which can make the images to flicker or make the colors shift if an electro-magnetic source approaches the screen or if an unshielded device approaches the screen. Finally, Chuang et al. [36] found that the glass envelopes of CRT devices contain toxic lead and barium, and that their phosphors can also contain toxic components such as cadmiums, which can cause environmental pollution.

LCDs were later applied to avoid the drawbacks of CRT, since the LCD displays have high energy efficiency, can be disposed conveniently, have multiple choices in regard to screen sizes, do not contain toxic elements, and do not suffer from magnetic interference. The LCD is a type of electronic visual display which employs the light modulating characteristics of liquid crystals, and the LCD needs a backlight source or reflector to emit light and display images [38]. The LCDs are applied in computer monitors, televisions, instrument panels, aircraft cockpit displays, signage, battery-powered electronic equipment, video players, gaming devices, clocks, watches, calculators, and telephones, and have substituted CRT displays in many applications [39].

However, the LCD display has some drawbacks, such as limited viewing angle, relatively low contrast, and reduced color saturation at low intensity levels [40]. The uneven backlights in some LCDs can lead to luminance distortion, particularly in edges of the screen. Most LCDs' backlights use pulse-width modulation (PWM) dimming technique to drive the display. But this design makes the LCD flash more seriously than a CRT display at 60 Hz refresh rate, since the LCD is lighted entirely in one field and then turned off, whereas the CRT screen is lighted by conscious scans within one filed [41]. The flicker of the LCD can lead to eye stress for some people. Finally, the quality of the LCD display in direct sunlight is very poor, and often not viewable. The LCD devices make improvements in regard to providing the natural light, but they provide dim light via the backlight technique and thus have limited application in outdoor environment.

For LCD devices, A/D converter is a necessity in the display device to transform the analog signal into a digital signal. This transmission process inevitably results in image distortion. Given the image distortion problem, DVI digital display technology is widely spread in the marketplace. Given the image distortion problem, digital visual interface (DVI) technology was proposed to avoid the distortion problem. The DVI is a video interface standard. DVI aims to enhance the quality of the PC display of the digital transmission. It is widely used in LCD, digital projectors and other display devices [42]. The DVI can transmit uncompressed digital video data to the display device. The DVI protocol interface can transmit the luminance and color signals of the pixels from the signal source (such as video) in binary mode to the display device. When the display device receives data signals of its native resolution, it can read numerical data of each pixel and display the data signals correspondingly. Analog approach to transmit pixel data is affected by adjacent pixels and electromagnetic noise and other data, which affects the data quality [43]. In the DVI interface transfer approach, each pixel in the register corresponds to a pixel in the display system, therefore ensuring the picture quality.

On the basis of the DVI interface, due to advances in digital media technology, people need a cable capable of transmitting audio and video signals simultaneously. High-definition multimedia interface (HDMI) is an interface for sending uncompressed audio and video signals, and can be applied in set-top boxes, video projectors, personal computer monitors, video games, integrated amplifier, and digital television equipment [44]. During the transmission, all the video data is encoded into data packets with HDMI transceiver chip using the "transition-minimized differential signaling" (TMDS) technology. TMDS is a differential signaling mechanism with which video pixel data can be encoded and transmitted via a serial connection. A HDMI display system includes a transmitter whose sources can be built-in graphics chips, and a receiver which is a circuit on the display and can receive digital signals, and decode and pass them to digital display circuit.

The video display systems containing LED devices have gained increasing popularity, as the improvement of LED techniques reduces the costs of LED devices. It is reported that the market for LED display has been having great success, and the sales price of displays drop annually, allowing larger displays to be built. In addition, the LED video display has advantages of energy efficacy and high contrast [45]. For these merits, video systems using LED modules have been become increasingly popular and LED video display industry has grown rapidly. Given such trends of LED display industry, many techniques to control LED display have been proposed. It is reported that the color calibration methods by software and hardware compensations can improve the image quality and color quality of the LED display. Kim et al. [46] proposed a dynamic nonlinear transformation algorithm to ensure that the LED display system can have high quality image in all kinds of ambient brightness. Chang et al. [47] showed an automated color inspection and compensation methods for color LED modules to improve the display quality.

The individual PWM converters can be used to control each LED, since although such converters are quite simple, they can control a large number of LEDs [48]. Similar approach is used in [49], and the LED drivers have been developed to include all dimming electronics on a chip, such as serial interface, PWM dimming, constant current LED drivers and even local dot correction. Up to date such drivers are expensive but are very attractive for future applications with the decrease of IC’s price. It is suggested that the LEDs and their control electronics are placed on same module in the integrated tile design, but the most expensive electronics is placed separately from the tile [50]. However, there are also challenges associated with the LED display system. The application of LED video display may impose certain complex circuitry for image presentation on screen. Nguyen [51] proposes that in the indoor display market, competing technologies can provide similar performance as LED displays, and that the resolution and view angle of LED displays need to be further improved in order to meet the requirements of the indoor environment. In addition, most LED display mainly focused on presentation of graphical advertisements or short movies, and therefore, may not be optimal for longer video display [52]. Tsai et al. [45] suggests that the field-programmable gate array (FPGA) can be used to control a 16 $\times$ 16 dots resolution tile by treating it as an independent microsystem, as can be seen in Fig. 4. As this design uses USB for data transfer, the maximum rate for data flow is limited to 1 MB/s. Also, the high definition video sources cannot be used in this kind of LED display system. The high definition video source is capable of generating 32 levels of each main color. Since this kind of LED display system uses conventional PWM dimming and thereby lacks color depth, it is not capable to control a large number of LED pixels.

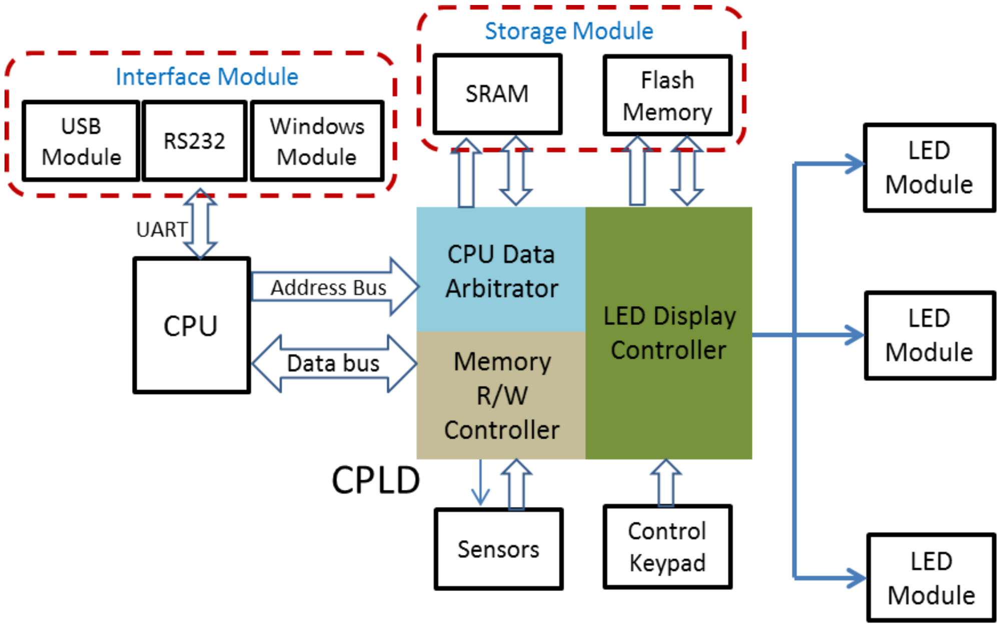


Figure 4 The block diagram of a LED display system [45].

1.2.2 Dimming methods for LED video display

With the widespread use of LED video display, there is an increased need for improvement of dimming techniques for LED video display to enable accurate control and regulation of colors in displayed images. So there has been a natural growth in demand for highly efficient and controlled LED drivers. Dimming capability of the LED drivers is important to sustain the energy efficiency scheme by an accurate control of the current.

Several studies were made on the driving methods for LEDs in the past. In general, they can be broadly divided into three main categories: (1) analog or direct current (DC); (2) ON-OFF pulse-width modulation (PWM); and (3) multi-level pulse-width modulation (MPWM). One of the methods is to control the amplitude of a continuous DC current (AM dimming technique), which can be achieved by adopting resistive dimming or external DC control voltage. A sinusoidal current dimming technique has also been used for driving LEDs with desired illuminations [54]. Analog or DC driving provides the highest luminous efficacy and the best color stability over dimming [55]. However, due to the intrinsic optical properties of LEDs, whose luminous output varies nonlinearly with the driving DC current, this driving method is suitable only for lighting applications that do not require linear dimming. The latter requirement is fulfilled when ON-OFF PWM driving method is used. By this approach, a constant DC current goes through the LEDs over the duration of the ON pulse, and the LEDs are fully turned off during the OFF duration. This way a chopped DC current is realized and the LEDs can be dimmed by varying the duration of the ON pulse, and a luminous output proportional to the ON duration is thus generated. In addition to the provision of linear dimming, ON-OFF PWM driving is also intrinsically compatible with the operation of large-area LED arrays since the brightness information embedded in the input digital video signal can be easily converted to the ON duration of the PWM driving signal without much computational effort. The drawback of the ON-OFF PWM driving method, however, is that it will lead to a reduced luminous efficacy of the LEDs.

The multi-level PWM driving method was recently developed as an extension to the conventional ON-OFF PWM [56, 57]. Instead of having two extreme states only, i.e. fully ON and fully OFF, the PWM driving signal is configured to switch between two current levels, with the lower current level being zero or non-zero. Depending on the required dimming level for the LEDs, multiple pairs of current levels can be adaptively selected, hence the name multi-level PWM driving method.

In the LED video display system, LED luminance control can be realized in two ways: one is the current intensity control method, and the other one is on-time control method [58]. The method of current intensity control can be achieved by adjusting the LED forward current in regard to different levels of luminance. For example, the LED forward current is divided into 256 levels, which means that the brightness can be divided into 256 levels. Luminous efficiency of this method is the highest. However, the current intensity control method is rarely used in practice because of its complicated operation and implementation.

On-time control method uses constant current to drive LED, and controls the conduction time per unit time to achieve different luminance levels. Since the LED has a fast response time, LED can be driven with a pulse light. For example, the pulse light with 1 MHz, 50% duty cycle and the peak current of 2 A has the same driving effect with the a driving current of 1 A theoretically in terms of brightness [59]. Therefore, adjusting the duty cycle of the current pulses can obtain different brightness levels. If the discrete data of each pixel of the video signal is used to control LED the conduction time, 256 full-color video image can be obtained. At the same time, the human eye has visual inertia. When the light pulse stimulation enters the human eyes, it takes some time for the image establishment and disappearance [60]. So the longer the conduction time is the more light-emitting energy the human eyes perceive. As long as the turning-on and turning-off frequencies of the LED display are properly controlled, the human eyes cannot distinguish the time period of non-emitting, i.e., no flickering. In the digital signal processing, time control can be easily obtained. So this study employs PWM method to control the luminance of the LED system, and makes improvement on this method.

For on-time control method, in the circuit design process, there are two methods to choose from: counter comparison and bit-plane method. In Fig. 5, to achieve the 256 full-color luminance, the time of one frame should be divided into 256 parts. The counter clock’s cycle time of each frame is 1/256 which is the minimum time period to turn on the LED display. Specifically, counter comparison method generates an initial value of 0 in the internal programmable logic device [61]. The maximum value is 256, and the counter can continue to increase the value in the clock control. Initial state of the LED display is set to turned-on. In the next clock cycle, the video data read from the memory is compared with the data set for counter. If the video data read from the memory is larger than the value of counter, the conduction time of the pixels is not enough. So the LED display should stay the turn-on state. If the video data is smaller than the value of counter, the conduction time of pixels meets the requirements. So LED display is switched from the turn-on state to the turn-off state.

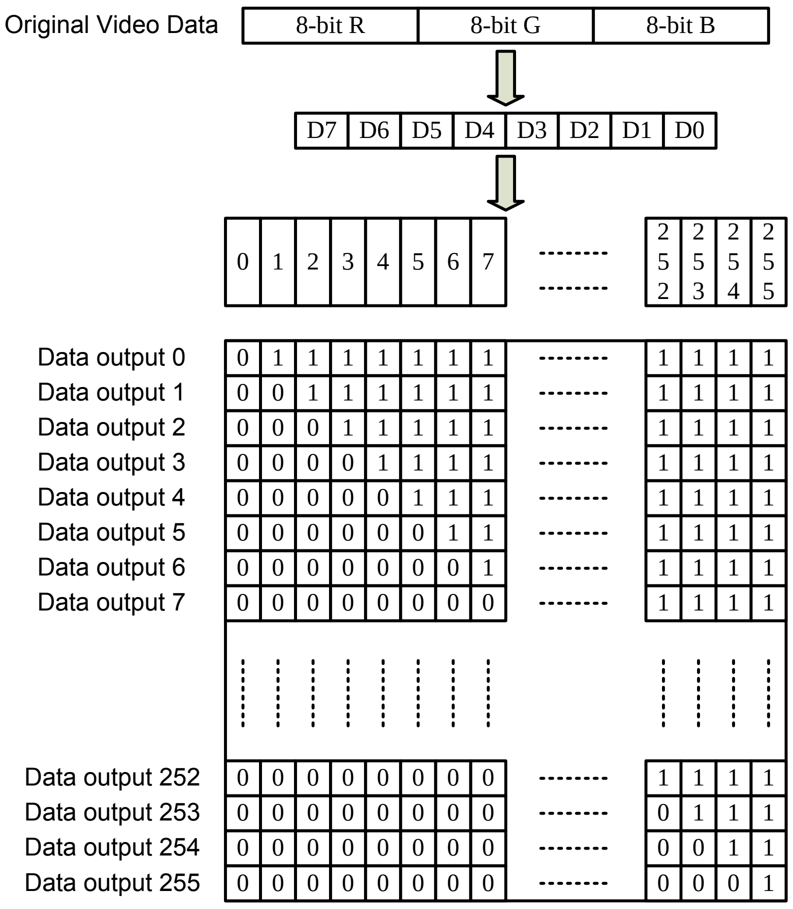


Figure 5 Conventional full-color video data dimming method.

In practice, however, this approach read the video data from memory and compare for 256 times in each frame. If the screen is slightly bigger, then a very fast read speed is needed to meet the design requirements [62, 63].

The full-color high-definition video signal has 256 brightness levels, and 8 bits digital signal can be obtained by decoding circuitry. In the LED display process, only 1 bit digital signal data needs to be displayed among the 8 bits (D7 - D0). So the signal can be displayed 8 times to present the entire 8 bits full-color video signal. The luminance value of each pixel can be controlled by adjusting the duty cycle, and the corresponding luminance value for each duty ratio is 8 bits. The duty-cycle value of the video data with 8 bits is shown as below. *T* represents the total time occupied by each byte, and *t* represents the conduction time of LED. Given that the control period is fixed, the different duty cycle control modules can be achieved by adjusting the value of *t* and the 8 bits video data. The conduction time given to the low brightness is short, while the conduction time given to the high brightness is long. Thus, a complete 8 bits duty ratio corresponding to the video data with 8 bits can be calculated by the sum of the value of "1" (turn on) times its duty ratio and the value of "0" (turn off) times its duty ratio, as shown in Fig. 6.

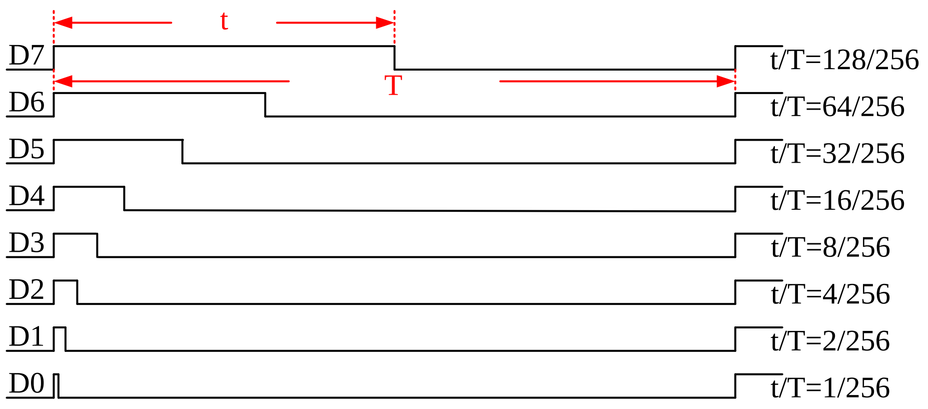


Figure 6 The explanation of bitwise separate method.

However, the conventional linear PWM dimming technique has one inherent disadvantage which is that in any pixels intensity, all the LEDs are switched on at the same time. Using this technique, a large number of LEDs may be lighted simultaneously in the LED display system, resulting in a large current spike on power supply which may lead to a serious electromagnetic interference (EMI) problem, as can be seen in Fig. 7.

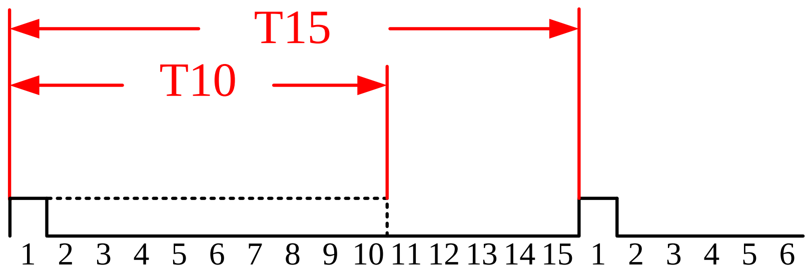


Figure 7 The explanation of linear PWM operation method.

On the basis of previous studies, some different weighted PWM dimming techniques are provided, such as the binary weighted PWM dimming technique [64], the discrete PWM dimming technique [65] and the bit angle modulation dimming technique [66]. This kind of dimming technique can reduce EMI problem by spreading the current spike into different time period. Here take binary weighted PWM dimming technique as an example, Fig. 8 compares the conventional PWM dimming technique and binary weighted PWM dimming technique.

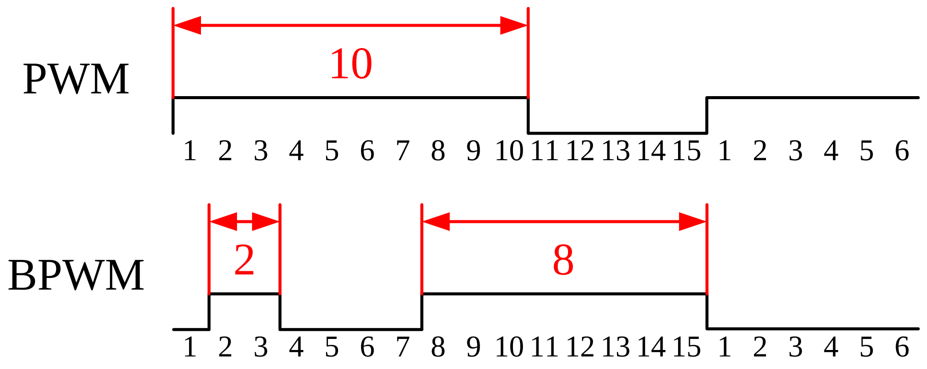


Figure 8 The comparison between PWM and BPWM with the same weight values.

A number of different pulses can be used by the binary weighted PWM dimming method for a single period. The width of each separate pulse is proportional to the weight of the bit corresponding to the code bit for the required luminous intensity [67]. The total pulses width for the binary weighted PWM dimming technique is equal to the conventional linear PWM dimming technique. The advantage of this dimming technique is to reduce the amount of required memory in control system and decrease EMI problems by spreading the current consumed.

However, in the conventional linear PWM dimming method and the binary weighted PWM dimming method, the minimum pulse depends on the data loading period. Thus, an improved binary weighted PWM dimming technique was provided with additional gating in Fig. 9. Such improved binary weighted PWM dimming technique is called gated PWM (GPWM) dimming technique [67]. The minimum lighting duration in GPWM dimming technique is much shorter than that for loading the video signal. Compared with the shortest PWM duration t_{load} defined by the video signal load time, the "on" period of LED is a short duration t_{min} of loading video signal operation. So the t_{min} can be used to replace the t_{load}. In such dimming method, the amount of PWM levels can be extended. Most of the dimming technique performance in binary weighted PWM dimming technique can be achieved when the GPWM dimming technique is combined with the BPWM dimming technique.

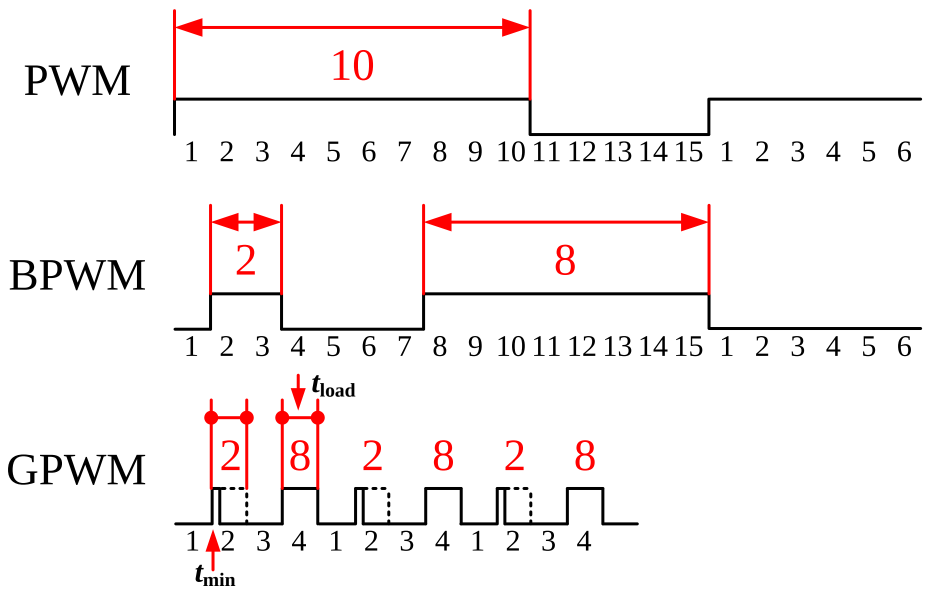


Figure 9 The comparison between BPWM and GPWM with the same weight values.

The third dimming method is derived from the combination of AM dimming method and PWM dimming method, and is called two-dimensional dimming technique. Its luminance intensity can be manipulated by varying the duty cycle and the current amplitude at the same time [55]. The typical techniques in this type of method, including pulse amplitude modulation (PAM) [58], bi-level PWM [56] and *n*-level PWM [57], are widely used to further improve the flexibility and accuracy on the control of luminous intensity of LEDs.

The first two dimensional dimming technique, pulse amplitude modulation (PAM) dimming technique was proposed for driving LEDs by Radiant [58]. In PAM dimming technique, both the duty cycle and the current amplitude are used to adjust the luminous intensity of LEDs. The PAM dimming technique has the advantage of choosing AM dimming technique or PWM dimming technique by varying the duty cycle and current amplitude. When the LED is driven under AM dimming technique, the duty cycle is set as 100% to control the amplitude of the current and to determine the luminous intensity of LED output. In the PWM dimming technique, the driver delivers constant and fixed current for LEDs, and the luminance intensity of LEDs depends on the duty cycle. So the major advantage of the two dimensional dimming technique is controlling the luminous intensity of RGB LEDs. In practice, PAM dimming technique is often used in AM driving mode. The luminance intensity can be adjusted by changing the current amplitudes, since the duty cycle is set at the same value. The problem in this case is because the current amplitude is varied. The accuracy and color stability of the current amplitude remains to be an issue. Meanwhile, when the duty cycle is not at 100%, the luminous efficacy of PAM dimming technique is lower than AM dimming technique.

However, the two dimensional dimming technique requires further improvement so as to control the duty cycle and the current amplitude at the same time. Some related works have been proposed to further improve the conventional two dimensional dimming techniques. Bi-level current driving technique [56] and *n*-level current driving technique [57] were proposed to further improve this driving method. The bi-level dimming method is a typical two dimensional dimming technique as it combines the DC offset with a PWM current. In this way, the bi-level PWM dimming method can improve the luminous efficacy. The main feature of this dimming technique is that two different levels of the current I and $I_{\max}$ are used to provide a lower and higher efficacy respectively. And the overall or average luminous efficacy for dimming processing in this method is higher than that in the conventional PWM dimming technique. The disadvantage of this dimming approach is that the minimum luminous intensity of LED is limited by the first current level. So in this case, this driving technique cannot be used in LED display system for its limited dimming ranges. Moreover, the *n*-level dimming technique is used based on the bi-level's advantage of higher luminous efficacy, as it plots the overall dimming range from minimum level to the maximum level into several bi-level sections. In this method, the luminous efficacy can approach the AM dimming technique as closely as possible. Even though LED dimming technique has advantages in efficacy and stability [68], its driver controller and driver circuit are difficult to design and cannot be implemented with low consumption. So this study aims to explore an effective dimming method with low cost and simple implementation, especially with dimming ranges which match video display system that features large dimming ranges.

1.3 Objective

This thesis mainly focuses on improving the energy efficiency of the RGB LED display system. Driving solution for improving the efficacy of the RGB LED display system is found. From the literature review, considering the merits of HDMI and the characteristics of LEDs, this study uses HDMI as the video playback unit for sending the video signals; the bi-level dimming technique and *n*-level dimming technique are the appropriate methods for the system. Specifically, a special case of practical implementation of *n*-level dimming technique can be employed to further improve the color stability and video resolution of the full-color LED display. This study will also commit to narrowing the gap by introducing the practical implementation of this special case PWM dimming technique based on the digital controlled RGB LED video display system. Moreover, the implementation of this technology only brings a very small increase in the hardware cost. Issues on color source data correction and dimming approach are also discussed.

1.4 Outline of the thesis

This thesis is organized as follows. Chapter 1 introduces the motivation of this study on the full-color LED video display. A literature review on the overview of video display and dimming methods in LED video display is provided. The potential improvement of full-color large-area LED display system is discussed. The research objective and outline of this thesis are also presented.

Chapter 2 describes the fundamental aspects of LED video display, including the video sources, LED display panel drivers and large-area sizes, followed with the architecture of a typical full-color LED video display panel system. It also presents a brief description of the basic concepts of the multi-level PWM dimming method and generalizes the design guideline for a three-level dimming scheme. And the algorithms of data correction on both the multi-level PWM dimming method and original video signal data sources are also discussed.

Chapter 3 proposes a modified energy saving full-color large-area LED video display system. The goal is to implement the multi-level PWM driving method with a minimum change to the architecture or operating principles of the existing systems. And it also presents the detail description of design and implementation of the LED video display in hardware and software. A prototype control system is built to verify performance of the proposed LED video display system.

Finally, chapter 4 gives a conclusion of the thesis. The major work and contributions are reiterated. Some suggestions for future research are given.

Chapter 2 LED video display and its dimming methods

In this chapter, the basics of LED video display will be firstly described, and the general issues associated with the conventional LED video display will be discussed. The system architecture of FPGA-based LED video display and its general operation will be discussed. Then the conventional dimming methods will be described, followed by the proposed energy-saving multi-level dimming methods. The related data correction problems will be discussed and the special issue of gamma correction between the differences of CRT and LED device will also be described.

2.1 Basics of LED video display

There are two categories of LED display panels: conventional display panels using discrete LEDs and surface-mounted device (SMD) panels.

Conventional panels using discrete LEDs are mostly used by outdoor LED screens. A bunch of red, green, and blue diodes are driven collectively to establish a full-color pixel with the square in shape. These full-color pixels are located with equal spacing, and their absolute pixel resolution is measured by the distance from center to center within the square-shaped pixels. Most indoor LED screens on the market use surface mounted devices (SMD) technology and there is a trend that SMD techniques are extended to be used by the outdoor screens. A SMD pixel includes red, green, and blue diodes installed in a single package, which is then installed on the driver board. The diodes are tiny and are set up in a compact manner. The difference between conventional panels and SMD panels is that the maximum viewing distance of SMD panels is 25% less than that of discrete diode screen with the same resolution [69].

In general, LED display system consists of data source transmission module, signal processing module, and LED display module. The basic operating principle of LED display is that the signal processing module receives the video signals from the data source transmission module and then presents them through the LED display module. LED video signals control the information required to decode circuits and control systems, and are obtained through the conversion process on the interface transmission. The driving circuit transmits signals required by the display to control the LEDs. Then video signals can be shown in the LED display panel. The structure of LED display system is illustrated as can be seen in Fig. 10.

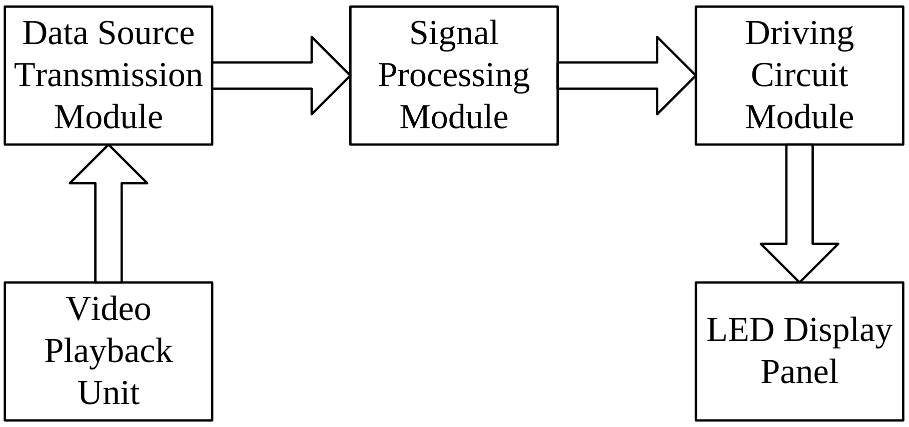


Figure 10 Architecture of basic LED video display system.

The LED driving process can be divided into two parts: transmission of the video data and display of the video data. The two parts are independent of each other. When the video data is transmitted, the video data should be simultaneously displayed. To achieve this object, the display of the video data requires a latch function. Thus, the video data transmission should achieve the function of serial-in and parallel-out; whereas the video display should achieve the function of the parallel latch. Therefore, when the video data has been input to the parallel latch display, the serial shift register can prepare for the next frame of video data, thus not impacting the current frame.

The smallest unit of LED display panel is the LED display module as can be seen in Fig. 11. Each module of LED display consists of two parts: the LED driver circuit and the LED display unit. Various circuits, including a signal transmission circuit, the input and output interfaces, and the display unit are assembled on a printed circuit board. LED is the smallest unit of a display module, and each display module generally contains 32 × 32 LED pixels. The video signals will be transmitted to LED display modules bit by bit. Each LED display module transmits the current video data to the next

LED display module and receives new video data from the previous LED display module until the entire video data is transferred completely. Signals are input from the rightmost of each line and output from the leftmost of each line to the next LED display module [70]. The information for each LED display module can be fully presented by repeating the above process. This process will continue until all the video data is transmitted. If the video data of one frame has been transmitted, the video information of this frame can be presented in the LED display screen.

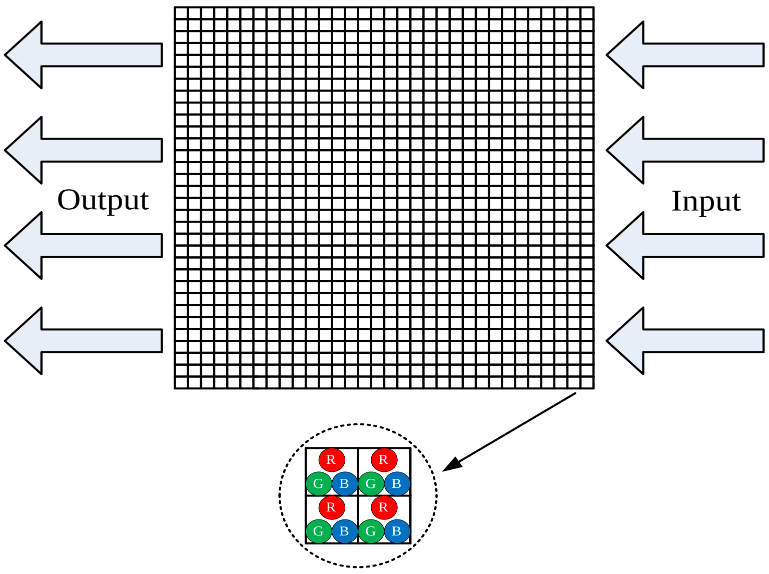


Figure 11 The smallest unit of LED display module which consists of RGB LEDs.

The RGB LED display system uses multiple connected LED display modules to present image directly. Due to great progress in computer and multimedia technology [45], the RGB LED display is widely spread for the advantage of its high visibility, sunlight visibility, long operation life, high reliability and resistance to the environment, low power consumption, low weight and full range of power, wide range of color spectrum, and flexibility in design. Fig. 12 gives an example of a full-color LED video display screen.

For the full-color LED display, the number of LED drivers is limited in each of the board, and each driver needs to control the RGB LEDs as many as possible at the same time. Large scale LED video displays market still holds competition: no other technology is capable to deliver large size at high contrast and durability. LED video displays manufacturers are offering new fancy size and shape products in attempt to explore the main advantage of LED displays over their competitors: the large scale combined with unbeatable luminance and contrast.

Figure 12 An example of a full-color LED video display panel shown outside the gym [20].

2.2 System architecture

The basic architecture of field-programmable gate array (FPGA) based LED energy-efficient video display control system, as can be seen in Fig. 13. The main operating principle of the system is briefly described.

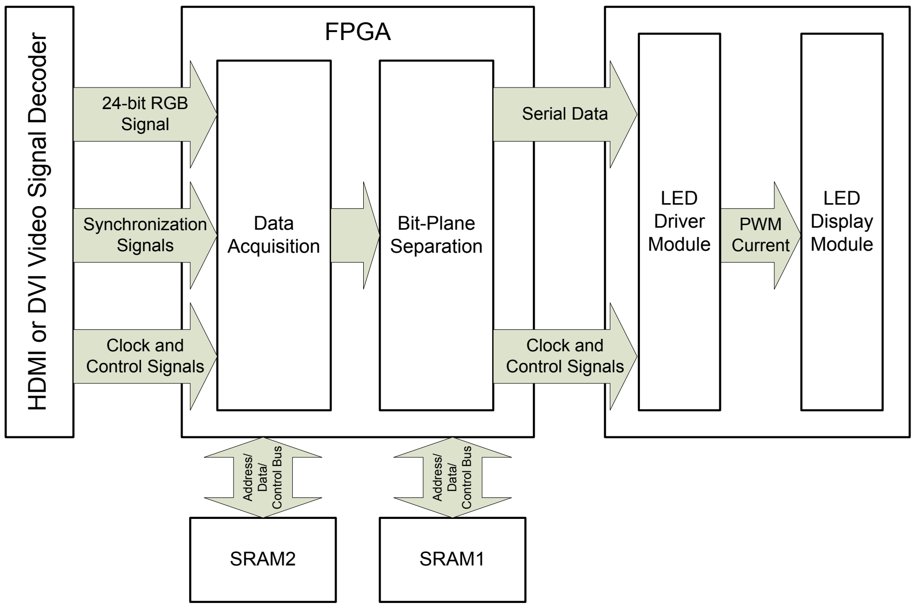


Figure 13 An FPGA-controlled LED display panel system architecture.

This system is composed of four main components, including the HDMI / DVI video signal decoding section, FPGA controller section, an external SRAM portion, and the LED display module section. The LED display cannot display HDMI or DVI video signals directly because of format incompatibility. So the HDMI video signals are firstly transmitted into 24-bit RGB signals. Each color (i.e., red, green, blue) contains 8 bits, thus enabling the presentation of 256 levels per color. The transmitted signals, together with the clock signals and the synchronization signals, are sent to the FPGA controller for further processing [53]. The primary synchronization signal includes output data enable (DE), the output data clock (DCLK), the vertical sync output (VSYNC) and a horizontal synchronization signal output (HSYNC). A typical example of the corresponding timing diagram is shown in Fig. 14.

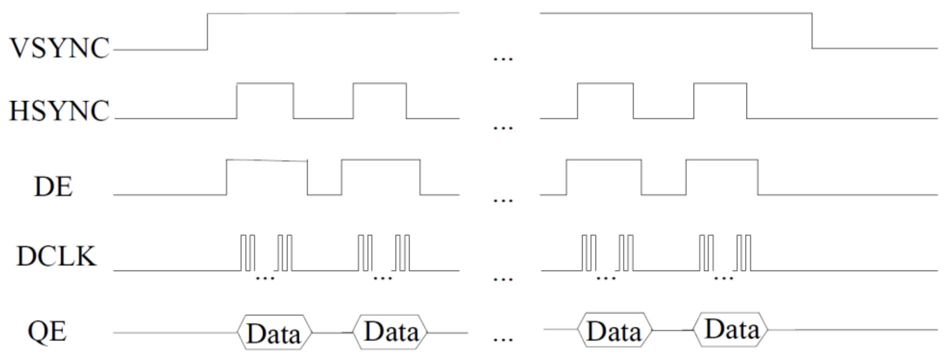


Figure 14 An example of timing diagram showing the various output signals generated by video decoder.

The rising edge of VSYNC indicates the start of a new field and it remains active over the time interval spanning from the start to the end of a scan. The rising edge of HSYNC indicates the start of a line scan (or row scan in the context of LED display panel) and it remains active over the time interval spanning from the start to the end of a line. Over the period of a row scan, the clock signal DCLK is used as a pixel counter to control the amount of data to be sent to the FPGA and LED display panel, depending on the panel size. For example, for an $m \times n$ panel, n clock pulses will be used to transmit the data corresponding to n pixels. The same purpose is achieved by using HSYNC as a row counter.

Within the FPGA controller, the incoming 24-bit RGB data are read into two external random-access memory (RAM) according to the page flipping (ping-pong buffering) approach. The minimum capacity of each RAM should be made sufficient to store the amount of data corresponding to one complete field, hence, for an m $\times$ n panel, the required memory space is $m \times n \times 24$ bits $= 3mn$ bytes. By adopting the ping-pong buffering approach, the two RAM are read and write alternately to ensure that the data flowing in from the video decoder is uninterrupted.

Subsequently, the data stored in these RAM are re-ordered to make them compatible with the operation of the LED driver module. The re-ordering mechanism, more commonly known as bit-plane separation, can be conveniently described using Fig. 15. With this approach, the data bits of all pixels are separated so that data bits having the same weight are collected into the same memory page. This simplifies the design of the LED driver module, as the data bits having the same weight should activate the LED pixels for the same duration of time. For example, if a time required for scanning a row is *T*, a bit-1 stored in the D7-plane will cause the corresponding LED pixel in the row to be activated for (128/256)*T*, whereas a bit-1 stored in the D0-plane will cause the corresponding LED pixel in the row to be activated for (1/256)*T*. Therefore, instead of converting data bits into the corresponding PWM duty cycles, which requires the use of D/A converters, the same total activation time can be achieved in a distributed or binary-weighted form. This has the advantage of avoiding simultaneous activation of all LED pixels in the same row, thus preventing a large transient load change to the onboard power supply.

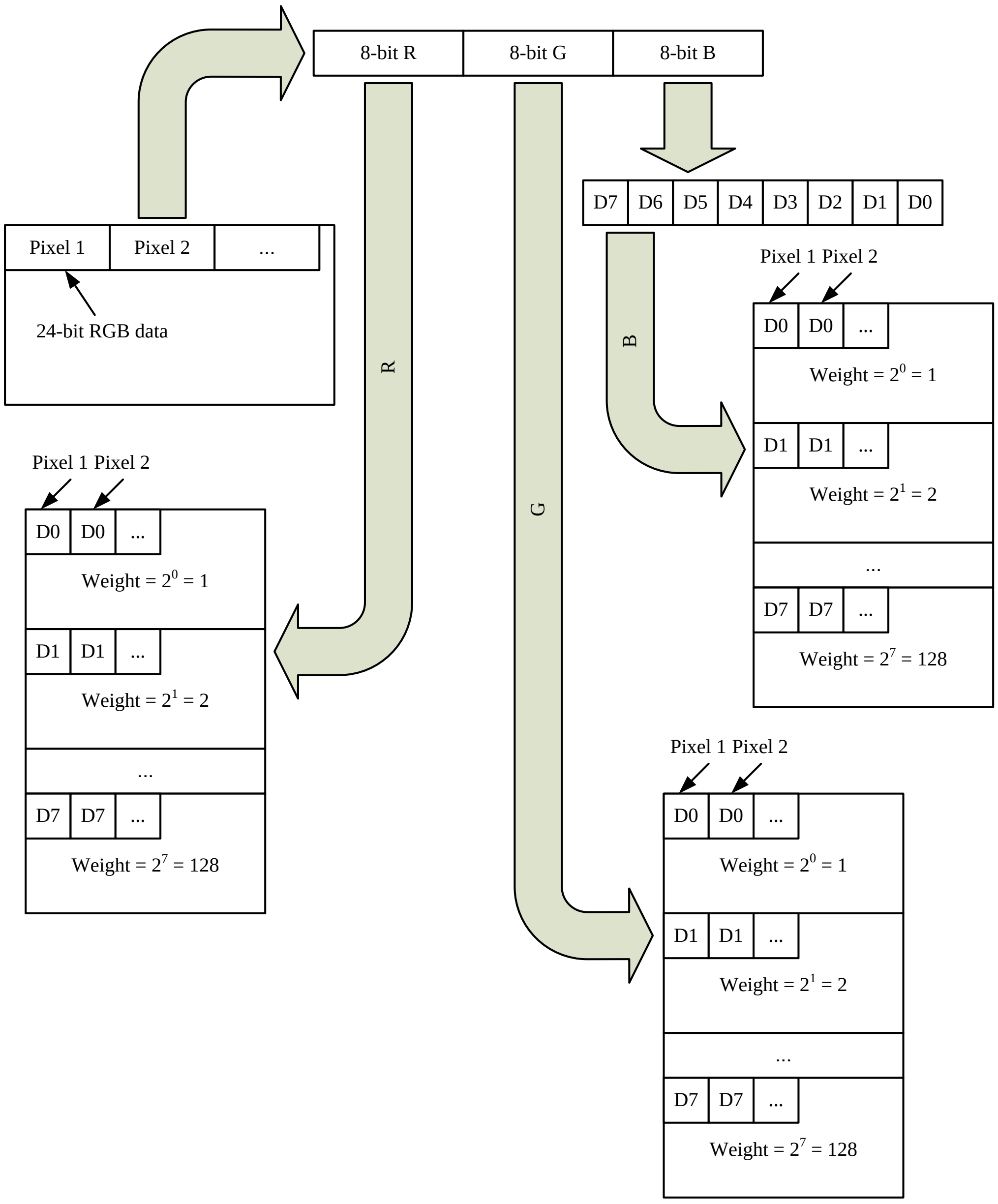


Figure 15 Separation of data bits according to their weights (bit-plane separation).

A typical LED driver module, as shown in Fig. 16, contains a 16-bit shift register that receives data transmitted serially by the FPGA controller. The FPGA is programmed to retrieve 16 data bits from the RAM per read-out and transmit them to the LED driver module in 16 clock pulses. LED display panels that have more than 16 pixels per row will require the cascading of multiple LED driver modules, and the FPGA will continue to transmit data serially until all shift registers are populated with data bits. When the transmission of all the data bits corresponding to one row is completed, LAT is raised to lock the transmitted data bits in the latches. The constant-current LED drivers are then enabled by lowering BLANK for a duration corresponding to the weight of the stored data bits. The same process is repeated until all eight groups of data bits (representing the eight different weights) are transmitted to the LED driver modules. The same process is repeated until all eight groups of data bits (representing the eight different weights) are transmitted to the LED driver module. After this the input value to the row decoder, ABCD, is incremented by one, and the transmission of the data bits for the next row will begin and the LED pixels will be activated similarly. A typical example of the corresponding timing diagram is shown in Fig. 17.

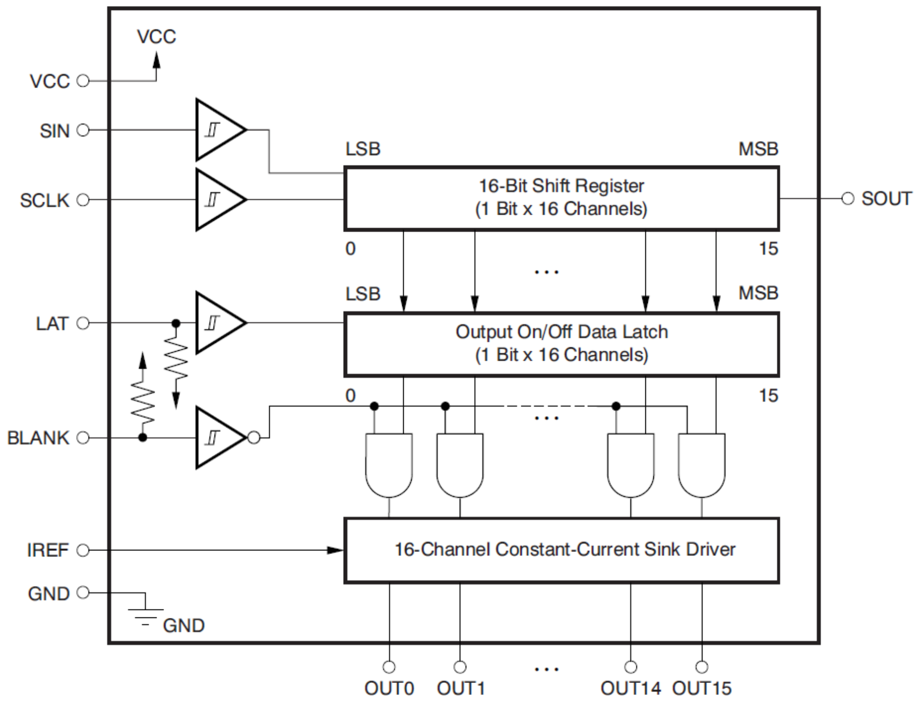


Figure 16 A typical LED driver module.

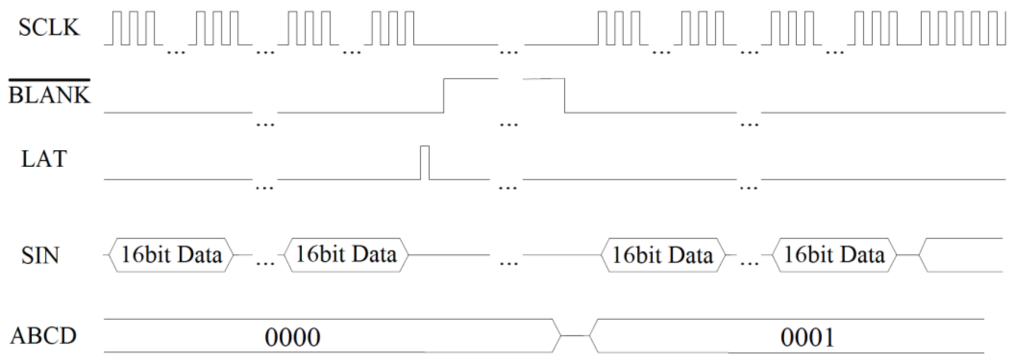


Figure 17 An example of timing diagram showing the various control signals applied to the LED driver module.

2.3 The proposed multi-level dimming method

The basic concepts of the multi-level PWM driving method will be discussed in this section, and the selection criterion for the mid-level current I_1 for a three-level driving scheme will be derived. Fig. 19 shows the typical current waveforms associated with the three driving methods discussed in the section 1.2.2, where DC, PWM, and MPWM is used to denote the analog, PWM, and multi-level PWM driving method, respectively. The current waveforms are drawn in such a way that they have the same average value equal to the value of the DC current.

For the MPWM driving method, the full dimming range of an LED, that is, 0 to I_2, is partitioned into multiple piecewise-linear sections (despite that only two sections are shown in Fig. 18), and each section is bounded by a pair of current levels, one higher and one lower current level [27]. When the required dimming level, or the equivalent average current, falls within a particular section, it is obtained by adjusting the relative time during which the higher current level is applied to the LED, thus resembling the duty-cycle-based dimming employed in the conventional ON-OFF PWM driving. In the illustrative example shown in Fig. 19, this is achieved by controlling the duration τ_2, which produces the average current as

$$I = \left(\frac{\tau_2}{T}\right) I_2 + \left(\frac{T - \tau_2}{T}\right) I_1 = I_1 + \left(\frac{\tau_2}{T}\right)\left(I_2 - I_1\right) \tag{2-1}$$

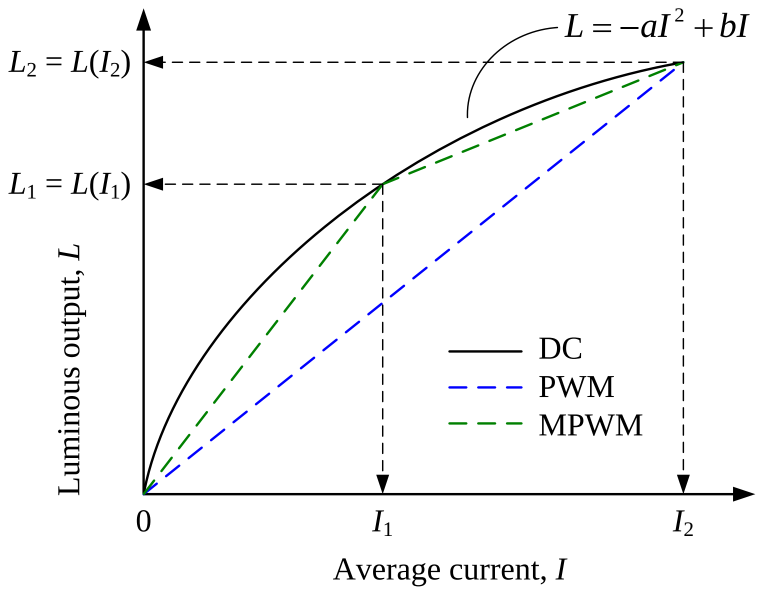
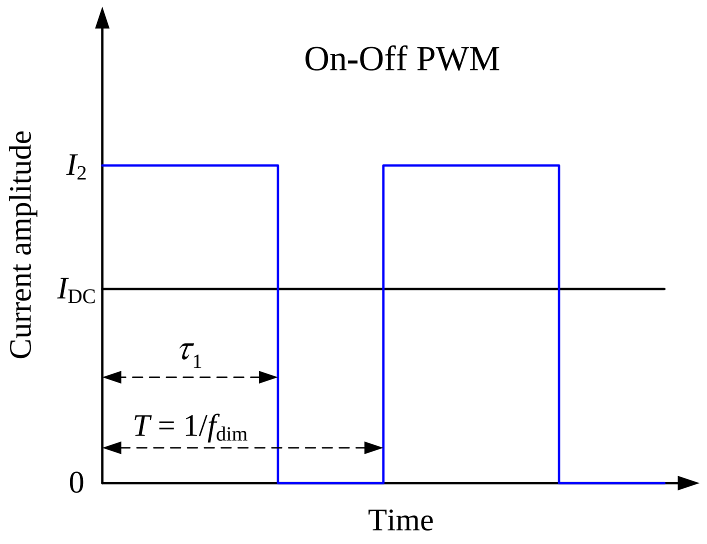
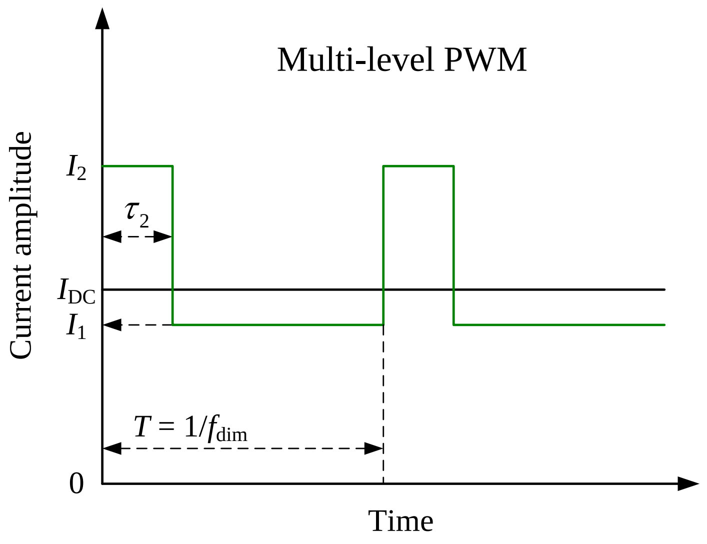


Figure 18 Dimming curves associated with three driving methods.



(a)



(b)

Figure 19   Typical current waveforms for three driving methods.

By referring to the illustrative example shown in Fig. 19, it can be seen that, if the required dimming level, or the equivalent average current, falls below I_1, the multi-level driving method will revert to the conventional ON-OFF PWM driving method, which is characterized by a PWM current waveform that switches between I_1 and 0. The main advantage of the multi-level driving method is that it offers a higher luminous efficacy in comparison to the ON-OFF PWM driving while retaining the characteristic of duty-cycle dimming [45], thus making it compatible with the digital operation of LED display panel systems. The conversion from an ON-OFF PWM driving signal to a MPWM driving signal, by keeping the average values of both driving signals equal, is straightforward, as shown below, which can be easily implemented on FPGA.

$$\left(\frac{\tau_1}{T}\right)I_2 = I_1 + \left(\frac{\tau_2}{T}\right)\left(I_2 - I_1\right) \quad \Rightarrow \quad \tau_2 = \left[\frac{\left(\tau_1/T\right)I_2 - I_1}{I_2 - I_1}\right]T \tag{2-2}$$

In theory, with the MPWM dimming method, the number of piecewise-linear sections can be infinitely multiplied in order to approach as closely as possible the DC dimming curve in order to maximize the luminous efficacy of LEDs, but doing so will cause the dimming action to become increasingly nonlinear. Therefore, a three-level driving scheme is suggested as a practical choice for achieving a compromise between dimming linearity and luminous efficacy.

Based on the proposed three-level driving scheme, it is now desired to derive the design criteria for selecting the three current levels employed. The selections of the highest and the lowest current level are straightforward. The highest current level I_2 is selected based on the specified maximum luminous output demanded from the LEDs, and the lowest current level is always set as zero to allow the LED to be completely turned off for achieving the maximum display's contrast ratio. Finally, the mid-level current amplitude I_1 should be selected with the aim to maximize the average luminous efficacy of the LED.

The notion of 'average luminous efficacy' is used here to reflect the fact that the luminous output of LEDs is continuously changing during the display of motion pictures and it is assumed that the LEDs will have equal probability of operating at any point on the dimming curve. From Fig. 20, the task is equivalent to maximizing the area of the shaded area (denoted by *A*).

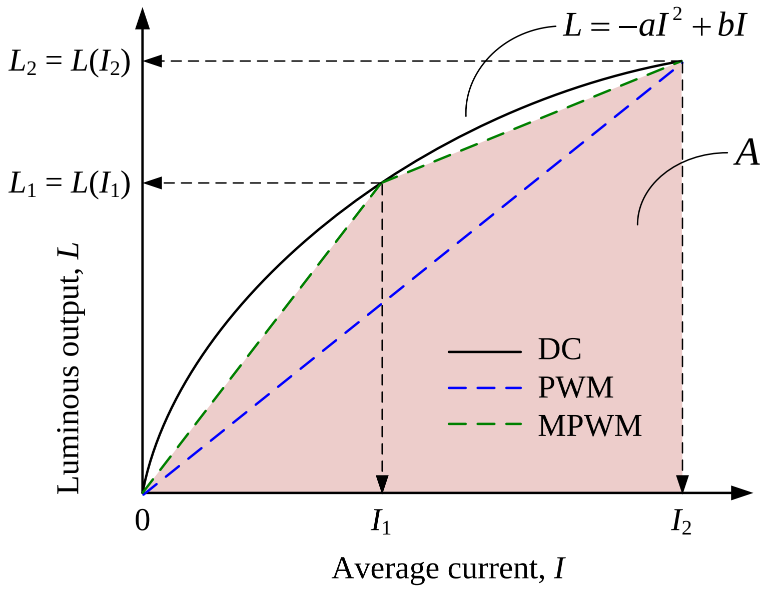


Figure 20 Maximization of area A leads to the maximum gain in luminous efficacy for MPWM driving over conventional PWM driving.

The area *A* is given by

$$\begin{aligned} A &= \frac{1}{2}L_1 I_1 + \frac{1}{2}(L_1 + L_2)(I_2 - I_1) \\ &= \frac{1}{2}L_1 I_2 + \frac{1}{2}L_2(I_2 - I_1) \end{aligned} \tag{2-3}$$

Since the luminous output of a typical LED is characterized by a concave shape, which can be fitted satisfactorily using a quadratic function of the standard form $y = -ax^2 + bx + c$, where, specifically, $c = 0$ since the luminous output of an LED should be zero when its forward current is zero, Eq. (2-3) can be further expanded as

$$
\begin{aligned}
A &= \frac{1}{2}\left(-aI_1^2 + bI_1\right)I_2 + \frac{1}{2}\left(-aI_2^2 + bI_2\right)\left(I_2 - I_1\right) \\
&= \frac{1}{2}a\left(I_2^2 I_1 - I_2^3 - I_1^2 I_2\right) + \frac{1}{2}bI_2^2
\end{aligned}
$$

(2-4)

Differentiating A with respect to I_1 and setting the resulting equation to zero gives

$$I_1 = \frac{1}{2}I_2$$

(2-5)

Since the result is independent of the coefficients of the quadratic function, it is equally applicable to all types of LEDs that exhibit a saturating luminous output characteristic. In fact, the result remains valid even for the case of $a = 0$, turning the quadratic function into a linear function $y = bx$, in which case the MPWM driving method will have the same performance as the conventional ON-OFF PWM driving method, and there will be no gain in luminous efficacy.

In the most general case, the selections of the number of current levels n and the individual values of the current levels $\{I_1, I_2, \ldots, I_n\}$ represent a complex optimization problem involving multiple constraints. One possible approach is to first maximize n subject to some customized constraints such as the additional cost of LED driver modules (see Chapter 3), the additional SRAM on FPGA for holding the sub-video signals (see Chapter 3), the available PCB area, the available output pins of FPGA and input pins of LED display panels for transmitting and receiving the sub-video signals, respectively, etc. Once the desired n is determined, the average luminous efficacy A is maximized subject to the constraints $I_1 < I_2 < \ldots < I_n < I_{\max}$, where

$$A = \frac{1}{2} L_1 I_1 + \frac{1}{2} \sum_{k=1}^{n-1} (L_k + L_{k+1})(I_{k+1} - I_k)$$

(2-6)

2.4 The algorithms of video data correction

In the previous section, the energy saving method has been discussed. At the same time, we need to ensure the display quality of the LED video display system. To this end, we proposed two algorithms to improve the video quality of the LED display. The first algorithm aims to correct the nonlinearity introduced by the MPWM driving method, whereas the second algorithm aims to correct the color distortion problem introduced by gamma correction.

2.4.1 The correction algorithm for MPWM

To account for the nonlinearity introduced by the MPWM driving method, and to ensure that color quality is preserved, additional calculations are required to estimate the average MPWM current required to produce the same luminance as the original PWM current by simple calculations based on the slopes of the two piecewise-linear sections. Consider the DC dimming curves of an LED which can be generally approximated by the quadratic form as shown in Fig. 21：

$$y = -ax^2 + bx$$

(2-7)

where x and y represents the equivalent PWM duty cycle (coded with 8 bits) and the normalized luminance, respectively. The coefficients a and b can be obtained by performing curve-fitting on the DC dimming curves obtained from the pulsating DC current experiment. Assuming that the original video signal commands a PWM duty cycle of x, and the corresponding normalized luminance is $y = x/255$, one can determine that the required MPWM current value is found in the lower piecewise-linear section y_1 if $y < y_1$ and the upper piecewise-linear section if $y \geq y_1$. Referring to Fig. 21, the equivalent PWM duty cycle of the MPWM current is denoted as x' .

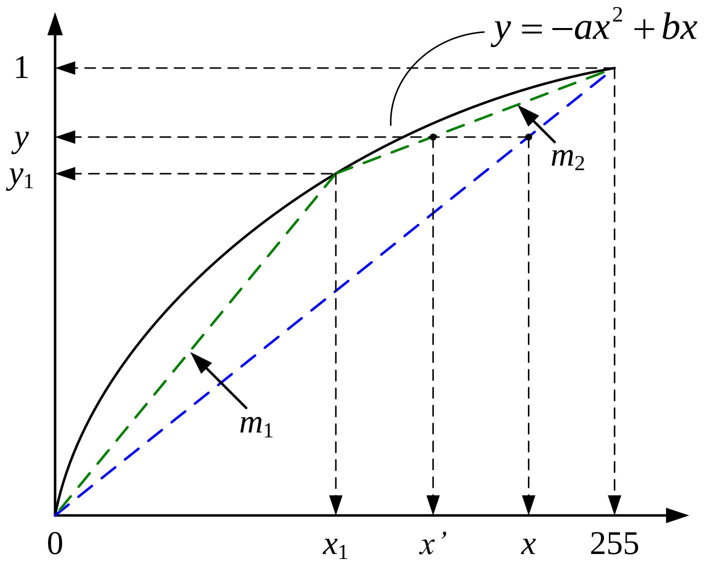


Figure 21 PWM to MPWM current conversion.

When $y < y_1$, the MPWM current will have an equivalent PWM duty cycle of

$$m_1 x' = \frac{x}{255}$$

$$x' = \frac{x}{255 m_1}$$

(2-8)

when $y \geq y_1$, the equivalent PWM duty cycle of the MPWM current can be calculated from

$$m_2(x' - x_1) + y_1 = \frac{x}{255}$$

$$x' = \frac{1}{m_2}(\frac{x}{255} - y_1) + x_1$$

(2-9)

where $x_1 = 128$ for realizing $I_2 = 2I_1$. If a small error is tolerable, the conversion can be performed more conveniently using one equation only given by Eq. (2-9), assuming that the DC dimming curve has been approximated very closely by the two piecewise-linear sections combined.

$$-ax'^2 + bx' \approx \frac{x}{255}$$

$$x' \approx \frac{b}{2a} - \frac{1}{2a}\sqrt{b^2 - (\frac{4a}{255})x}$$

(2-10)

2.4.2 The anti-gamma correction algorithm

There are no specific image signals and standards for full-color LED video display, since the video sources are mostly designed for the CRT display. In the LED video display system, the luminance of LED is linear to the driving signal, as the gray scale of LED is controlled by the lighting time or the lighting intensity. So the LED display can be activated with linear video signals. However, the video sources output by mainstream CRT monitors are non-linear, since they are processed by the gamma correction. Gamma correction, also known as gamma compression or encoding, can be used to encode the video signals to match the non-linear requirements of CRT display devices. So if these signals are directly send to the LED display, the gray scale of the LED display will have distortion problems. To alleviate the distortion issue, the anti-gamma correction will be conducted before the video signals are sent to the LED display, thus improving its video quality.

In conventional signal transmission system, to get high-quality images, signals from the camera in the transmission process should be linear, that is image luminance is proportional to the object brightness. However, luminous brightness of the CRT tube is not proportional to the voltage of control signals, but proportional to its power of gamma value. So the video sources should be processed with gamma correction prior to transmission of the video signals. A typical example of the characteristics of CRT performance is shown in Fig. 22.

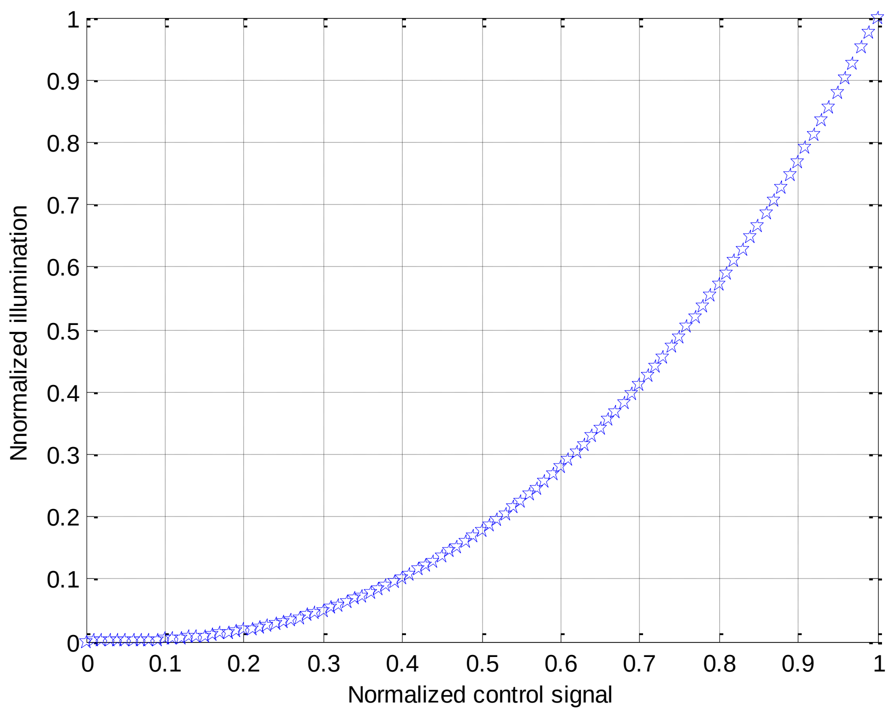


Figure 22 The relationship between the normalized control signal and normalized illumination in CRT display.

According to the CRT emission characteristics, the relationship between the brightness and the control voltage is not linear. The corresponding equation is

$$L_{\text{out}} = (L_{\text{in}})^{\text{Gamma}} \tag{2-11}$$

L_{in} : the input value of the original video data.

L_{out} : the result after the gamma correction of the output value.

The gamma value is different depending on the equipment, and the value generally ranges between 2.0 and 3.0. In this study, the gamma value is set as 2.5. Since this distortion is caused by the hardware implementation, anti-gamma correction is implemented by the software to solve this problem.

However, the luminance intensity of LED display is mainly determined by its current intensity and its external voltage. When the external voltage is constant, the strength of current is constant through the LED, and the luminance intensity of LED display also remains constant. Because the emission time of the LED light is less than the residence time of the human eye, the human eyes can view continuous images of LED display. When the current is constant, the gray scale level of the video source is completely determined by the current pulse. LED display brightness is linear to the current pulse width, so the brightness of the LED display is proportional to video image signal. The equation is shown as below:

$$L_{\text{out}} = L_{\text{in}} \tag{2-12}$$

L_{in} : the input value of the original video data.

L_{out} : the result after the gamma correction of the output value.

Full-color LED display is driven by the digital signal. The gray values are controlled by the duty cycle of corresponding LED pixels. So the gamma value is set as 1. The relationship between the brightness and the control signal in the LED display is linear. A typical example of the characteristics of LED performance is shown in Fig. 23.

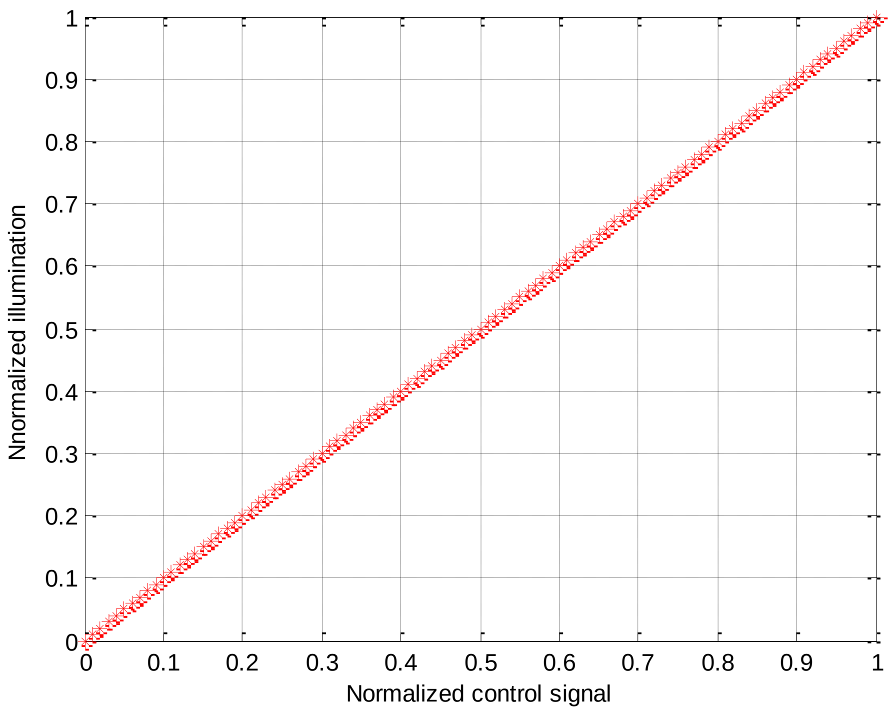


Figure 23 The relationship between the normalized control signal and normalized illumination in LED display.

The corresponding correction equation of anti-gamma is as follows:

$$L_{\text{out}} = ((L_{\text{in}})^{\text{Gamma}})^{\frac{1}{\text{Gamma}}} = L_{\text{in}}$$

(2-13)

L_{out} : the output signal of full-color LED display

L_{in} :the original image to be displayed.

$(L_{\text{in}})^{\text{Gamma}}$: the input value of the original video data with gamma correction.

There are two methods to conduct gamma correction. The first method is the real-time arithmetic function which obtains more accurate values than a look-up table (LUT), since a well-founded formula can be used to calculate the gamma correction block. However, it has complicated logics for arithmetic calculation block due to the use of floating points. So, it is not suitable to be used to process HDMI video sources. The other method is the LUT that saves the corresponding function for the correction in RAM. The LUT can check the input values by matching a list of valid items in an array. Also, it can avoid the complex calculation of power and floating points so that it can process the video data more quickly than the real-time arithmetic function.

As can be seen in Fig. 24, the red line represents the control signals of the LED display. The blue line represents the signals processed with gamma correction that are output by the CRT monitor. The green line represents the signals used in first method using real arithmetic function. The red line represents the signals used in the second method using LUT algorithm that the corrected control signals written previously in RAM.

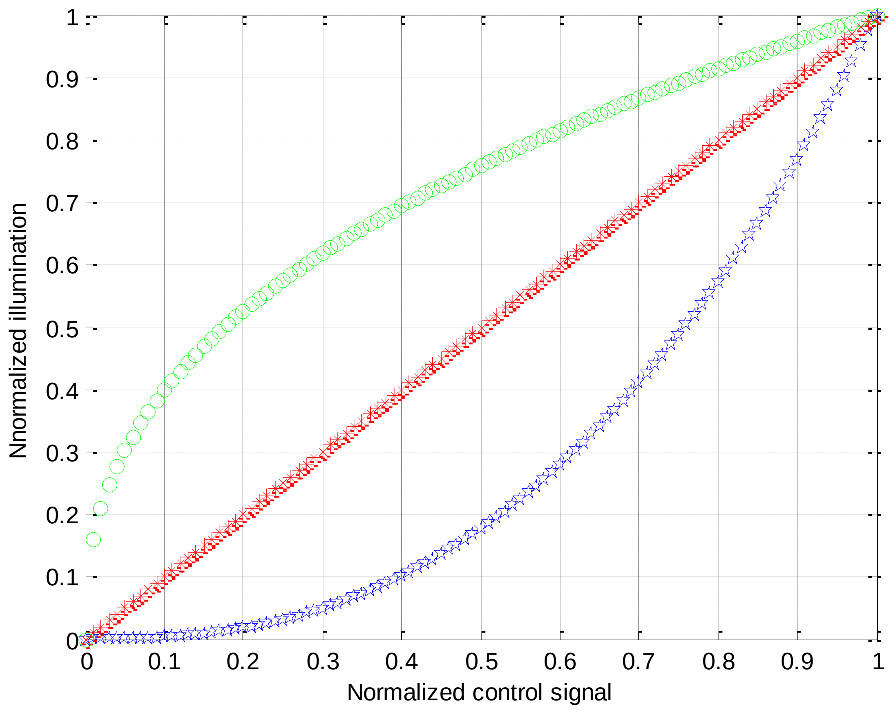


Figure 24 Anti-gamma correction.

Hence, according to the two data correction algorithms (e.g., the correction algorithm for MPWM and the anti-gamma correction algorithm), when the FPGA reads the original video signals from the video signal decoder, it will convert the signals to the corresponding data. This process will be accomplished before the signals are passed to the data-splitter for further processing.

2.5 Summary

In this chapter, the basics of LED video display have been introduced and followed with the related general requirements about such kind of display panel system. The FPGA based LED video display system has also been proposed with details about the system architecture, the display modules, the dimming techniques and the design of driving circuit of the LED video display. Then, the conventional AM and PWM dimming methods have been presented. To decrease the power consumption of the LED video display system, its luminous efficacy should be improved. So a multi-level current driving method has been proposed theoretically. Based on the physical characteristics of LEDs, this study proposes that the luminance efficacy of LEDs can be enhanced by introducing a dc-offset component into the PWM current [68]. Furthermore, the proposed multi-level current driving method can integrate the features of conventional techniques, in which both current levels and duty cycle are adopted to manipulate variables. The design criteria for selecting the three current levels employed has been derived. The highest current level is selected based on the specified maximum luminous output demanded from the LEDs; the lowest current level is set as zero; the mid-level current amplitude should be selected with the aim to maximize the average luminous efficacy of the LED. Finally, the algorithm of anti-gamma correction has been introduced to accommodate the video images used in CRT display so that they can also be well employed by the LED display.

Chapter 3 LED video display design and implementation

In this chapter, the modified LED video display system architecture will be firstly proposed. Then the hardware implementation will be provided, and the energy saving dimming method will be applied in the hardware assembly of the proposed LED display panel system. In the section of software implementation, the architecture and the simulation result of FPGA controller will be provided. Moreover, the experimental results on energy saving methods will be presented, and the verification of the LED display system performance will be given. Finally, both the hardware realization and the software realization on the large-area LED display system will be proposed.

3.1 Modified system architecture

In comparison with the existing system architecture, the modified system includes a data splitter that divides the incoming video signals into two sub-video signals as shown in Fig. 25. The algorithm based on which this is achieved has been described in Chapter 2.

The two sub-video signals are individually subjected to the process of bit-plane separation and output serially to the two LED driver modules. When the LED pixel's current is less than half of the maximum value, i.e. I_1, only one of the two driver modules will be activated, whereas the other driver module will be deactivated by sending a null signal to it, i.e. [0000 0000]. When the LED pixel's current exceeds I_1, one of the two driver modules will be always on by sending a full signal to it, i.e. [1111 1111], whereas the other driver module will deliver a PWM current to the LED pixel. This way, the currents from the two driver modules will combine to form a three-level current waveform comprising of 0, I_1, and I_2 (see Fig. 19).

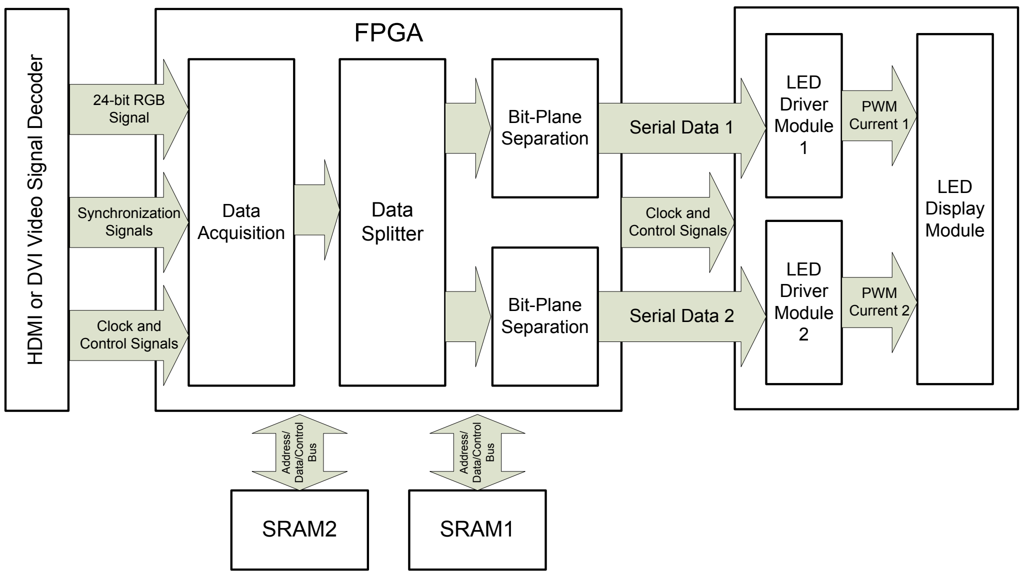


Figure 25 Block diagram showing the modified system architecture.

The advantages of this implementation include the following:

1) The MPWM driving method, which can result in a significant energy saving in large-area LED display systems, can be realized;

2) The data-splitter can be implemented on the FPGA without incurring additional cost compared to existing systems;

3) Although the number of LED driver modules is doubled in the modified system, the additional cost is not doubled given that IC manufacturers can redesign their ICs to house two or multiple such modules within the same IC package.

Although the MPWM driving method can be implemented with an arbitrary number of current levels, a three-level driving scheme is adopted in this work, with the lower current level I_1 chosen to be 50% of the higher current level I_2, according to the justification given by Eq. (2-5). With this configuration, the MPWM driving method can be implemented on FPGA with the minimum additional computational effort, thus avoiding the need for more powerful and costly hardware or performance degradation of the system due to additional computation.

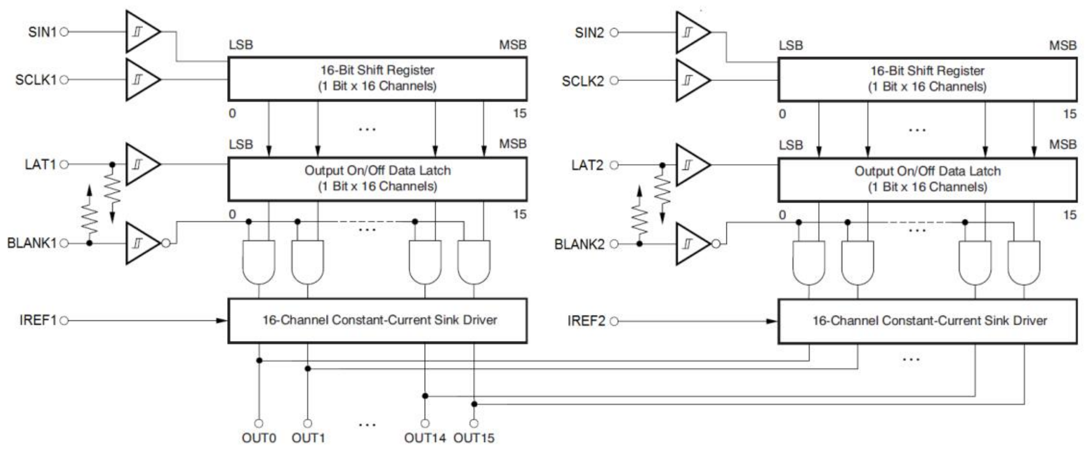

SIN1
SCLK1
LAT1
BLANK1
IREF1
LSB
MSB
16-Bit Shift Register (1 Bit x 16 Channels)
0
15
Output On/Off Data Latch (1 Bit x 16 Channels)
16-Channel Constant-Current Sink Driver
SIN2
SCLK2
LAT2
BLANK2
IREF2
OUT0
OUT1
OUT14
OUT15

Figure 26 Implementation of three-level MPWM using two LED driver modules connected in parallel.

To realize a three-level MPWM dimming method on LED display systems, two LED driver modules are linked in parallel to offer two constant current drivers, as shown in Fig. 26. Each of these drivers is configured to deliver a maximum current of I_1 to the LED pixels. Hence, the use of two LED driver modules enables a maximum current of $I_2 = 2I_1$ to be delivered to the LED pixels. With two driver modules, two separated bit-streams are needed to determine the average current delivered to each LED pixel. The method for converting the original 8-bit video signal into two separate bit-streams will be discussed in the next section.

The proposed structure can be similarly expanded for MPWM driving scheme employing more than two current levels. Should the MPWM driving method find recognition and adoption by the LED display panel industry in the future, integrated circuits containing multiple driver units can be fabricated for cost saving and compactness. A significant energy saving is therefore made possible at the expense of a small increment in hardware cost.

Since two LED driver modules are employed, two separate sets of data bits transmitted from the FPGA are needed to control them. For each color, this requires the conversion of the 8-bit video signal into two 8-bit sub-video signals for driving the two LED driver modules such that the combined current is equal to the required current represented by the original 8-bit video signal. The conversion can be easily illustrated using the following example.

Consider that the required intensity level of a given LED pixel (per color) is represented by the 8-bit video signal: [1010 0000]. If a single LED driver module with IREF = I_2 is employed, the average current delivered to the LED pixel will be:

$$\begin{aligned} I_{\text{pixel}} &= \left(\frac{2^7 + 2^5}{255} \right) I_2 \\ &= \left(\frac{160}{255} \right) I_2 \\ &= \left(\frac{320}{255} \right) I_1 \quad \left(\because I_2 = 2I_1 \right) \\ &= \left(\frac{255}{255} + \frac{65}{255} \right) I_1 \end{aligned} \tag{3-1}$$

The last line of Eq. (3-1) indicates that the same average current can be obtained using two LED driver modules with IREF = I_1, one of which is fed with the binary equivalent of 255/255 = [1111 1111], and the other with the binary equivalent of 65/255 = [0100 0001]. The two resulting 8-bit sub-video signals will be subject to bit-plane separation as in the conventional systems, where they will be mapped onto different memory pages and subsequently transmitted to the two LED driver modules.

Based on the example discussed above, it can thus be generalized that if x is the decimal equivalent of the 8-bit video signal for an LED pixel (per color), the video signal can be converted into two 8-bit sub-video signals consisting of the following:

Case A: [1111 1111] and [the binary equivalent of $2x - 255$], if $x \geq 128$, or

Case B: [0000 0000] and [the binary equivalent of $2x$], if $x < 128$.

Since $x = 128$ corresponds the original 8-bit video signal having the MSB = 1, the value of the MSB can be used to distinguish between the two possible cases stated above, and hence the conversion can be executed very efficiently without having to repeat the calculation steps shown in Eq. (3-1).

3.2 Hardware implementation

This system uses the HDMI interface to transmit video sources. The HDMI decoding circuit is the receiving component in the LED video display system, and it can get digital signals from the HDMI graphic interface. According to the transition-minimized differential signaling (TMDS) protocol, the HDMI decoding circuit sends decoded signals to the control system to format video signals. Since HDMI decoding circuit sends the output of differential TMDS signals, it requires a special decoding circuit to decode and store the video signals. The video decoder chip TFP401A, specifically used by flat panel display (FPD) as a digital receiving chip, plays an important role in decoding video signals.

Architecture of video source decoding circuit design based on the HDMI interface is shown as the Fig. 27. AT24C02 in the figure is used to store extended display identification data (EDID) of the HDMI display device. AT23C02 is the I2C serial bus memory for storing EDID signals and it through the DDC channel to achieve plug and play feature of the display device. When the power is on, AT24C02 transmitted the EDID data to PC through the display data channel (DDC) under the control of SCL clock. After reading EDID data, PC performs the identification and configuration. Then TMDS links will be activated. In the Fig. 27, RX2 and RX2+ denote red differential signals; RX1- and RX1+ denote green differential signals; RX3- and RX3 + denote blue differential signals; RXC- and RXC + denote differential signals of the clock. After conversion, TFP401A can output the luminance values of R[7..0], G[7..0] and B[7..0] to FPGA. The data clock is DCLK, and the signals of DE and line / field and the signals of VSYNC / HSYNC can be transmitted to FPGA controller.

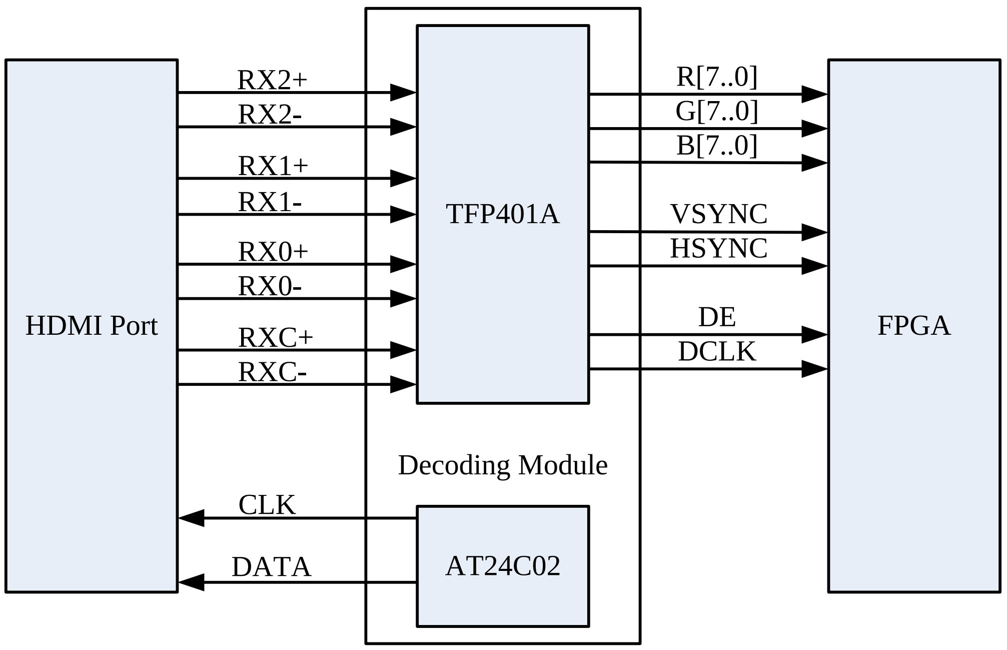


Figure 27   Architecture of video source decoding design.

In the LED video display system, the controller device is the core of the whole system. In order to obtain maximum flexibility, a programmable logic device can be used. The selection of programmable logic device mainly depends on the number of logic elements (LE), the number of embedded RAM, the requirement on the speed, the number of user input/output (I/O), the requirement for phase locked loops (PLL), and internal production device, etc. For embedded RAM, this design uses many internal RAMs to create a dual-port RAM that is mainly used in the bit plane separation module. The amount of RAM used in the bit-plane separation is 256 × 24 × 16 × 2 = 196608 bits, so 196608/1024 = 192 embedded RAM Kbits are needed. Considering the phase-locked loop, this study uses clocks with one frequency: 20 MHz. This design is generated by one phase-locked loop. Considering the IO number, 32 IO pins are needed to decode the HDMI. Two SRAM require

88 IO pins in total. 16 × 2 × 6 = 192 IO pins are required for the LED display driver signals, so a total of 312 IO pins are needed. Based on the above analysis, EP3C55, EP3C80, EP3C120 can meet the system's requirement. To fulfill the requirement for large-scale LED video display system (discussed in section 3.5), this study chooses Altera Cyclone EP3C120 FPGA, and the related parameters are shown in table 2.

Table 2 Comparison of Cyclone III FPGA family.

Device	EP3C5	EP3C10	EP3C25	EP3C55	EP3C80	EP3C120
Logic elements(LEs)	5,136	10,320	24,624	55,856	81,264	119,088
Embedded memory(Kbits)	414	414	594	2,340	2,745	3,888
Embedded multipliers	23	23	66	156	244	288
Phase locked loops(PLLs)	2	2	4	4	4	4
Maximum user I/O pins	182	182	215	377	429	531

In order to simplify the circuit structure, reduce the cost of the display, and improve energy efficiency of the system, the thesis design a full-color LED video display system using the way of 1/16 scan, a typical circuit of the corresponding driver circuit is shown in Fig. 28. Each LED dot matrix includes 32 × 32 pixels. Using common anode design approach, each line in the display modules has 8 dot matrix modules which contain 256 LEDs. These LEDs are controlled by four to sixteen decoder 74HC138D and SI4953 driver chips. Since the output of 74HC138D is low and SI4953 is a driver for low output, the scanning process is determined by four lines. The column driver cathode LED uses constant current driver MBI5026, and the output current can be adjusted by the external resistor. Due to the use of constant current driver design, the LED luminous intensity is proportional to the PWM duty cycle [27], which achieves LED brightness control. The circuit diagram of modified LED video display module is similar to the conventional one, but it adds one more constant current driver MBI5026 as can be seen in Fig. 29. The external resistor (R_{ext}) can determine the output current of each channel (I_{out}). The link between I_{out} and R_{ext} is given in Fig. 30. The output current can be given from the following equations:

$$V_{\text{r-ext}} = 1.26\,\text{V} \tag{3-2}$$

$$I_{\text{out}} = (V_{\text{r-ext}} / R_{\text{ext}}) \times 15 \tag{3-3}$$

R_{ext} donates the resistance of the external resistor that is connected to R-EXT terminal, and $V_{\text{r-ext}}$ donates the voltage of R-EXT terminal. The magnitude of current (as a function of R_{ext}) is around 20 mA at 945 Ω in the conventional LED video display module and 10 mA at 1890 Ω in the modified LED video display module. The specifications of the proposed LED display module circuit can be seen in table 3.

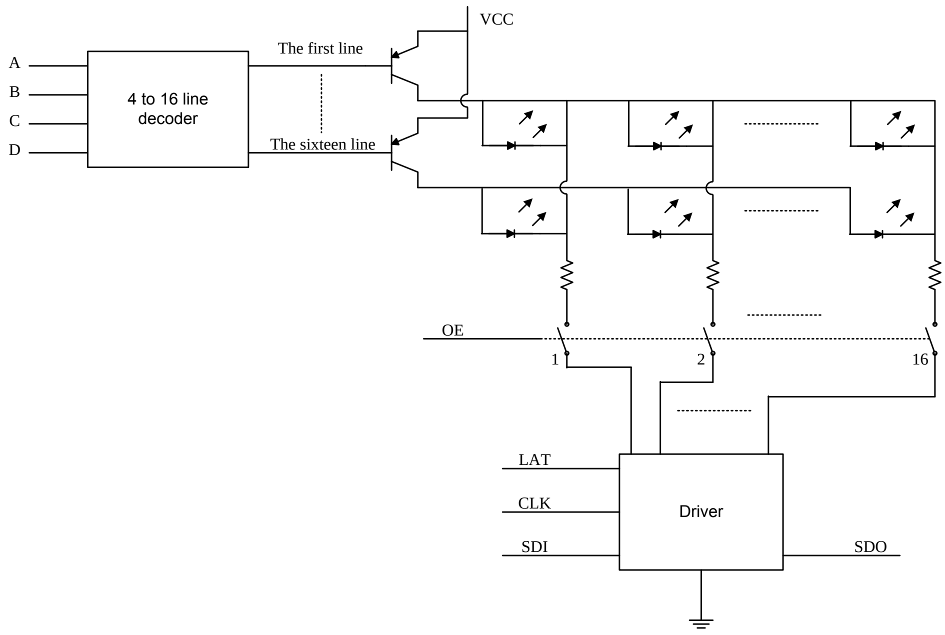


Figure 28 Circuit diagram of conventional LED video display module.

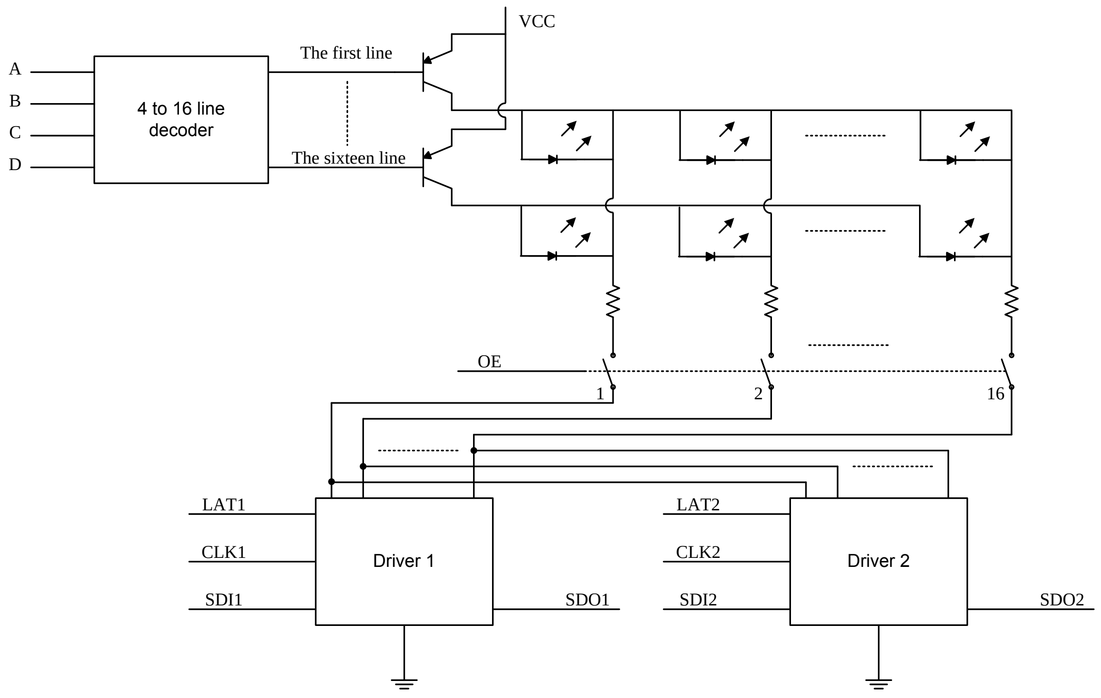


Figure 29 Circuit diagram of modified LED video display module.

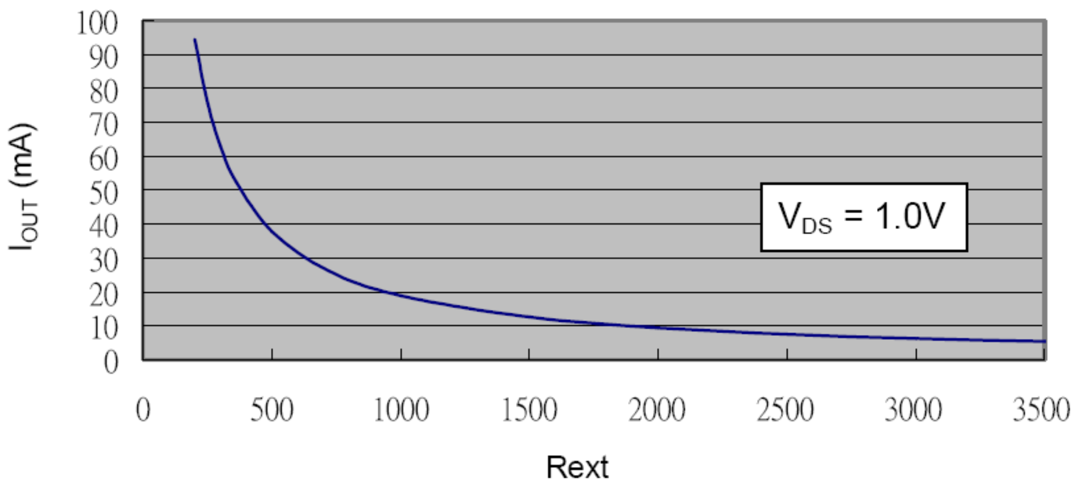


Figure 30 Resistance of the external resistor.

Table 3 Specifications of the LED video display module circuit.

Parameter	Symbol	Nominal value
Input voltage	V_{in}	5 V
Diode voltage	$V_{o,\,R}$	2.0 V
	$V_{o,\,G}$	3.2 V
	$V_{o,\,B}$	3.2 V
Current levels	I_0	0 mA
	I_1	10 mA
	I_2	20 mA
Maximum average output current	I_{max}	20 mA
Current sense resistors	R_R	33 Ω
	R_G	0 Ω
	R_B	0 Ω

Since each pixel matrix contains 24 bits video data information according the above analysis, the video data for the screen should be: 1920 × 1080 × 24 = 55296000 bits in order to present all the information of the computer screen on the LED video display system. If HDMI high-definition video data is used and the screen is refreshed 60 times within one second , then high definition video data source within a second will reach :1920 × 1200 × 24 × 60 = 3317760000 bits = 3164 Mbits. The resolution of the LED video display system in this paper is 256 × 192, that is the image with 256 × 192 pixels on the computer screen will be displayed on the LED video display system. As can be seen in Fig. 31, the amount of data within one frame of the video source is 256 × 192 × 24 = 1179648 bits, and the amount of data within one second is: 256 × 192 × 24 × 60 = 70778880 bits = 67.5 MHz.

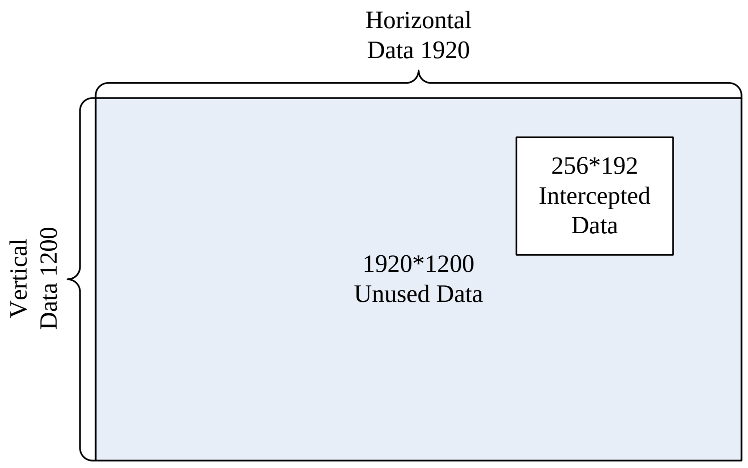


Figure 31 Interception of video source.

The front view and the back view of a single LED module consisting of 32 × 32 LED pixels can be seen in Fig. 32. The structure of the prototype energy saving LED video display panel is shown as Fig. 33. The LED display panel is composed by 48 LED display modules with 6 rows and 8 columns, and the resolution of the LED display is 256 × 192. The signal is input from the rightmost module in each row and output from the leftmost module of each row. Signals are input with six rows together from the rightmost modules, so the data from the computer screen can be fully displayed on the LED display screen.

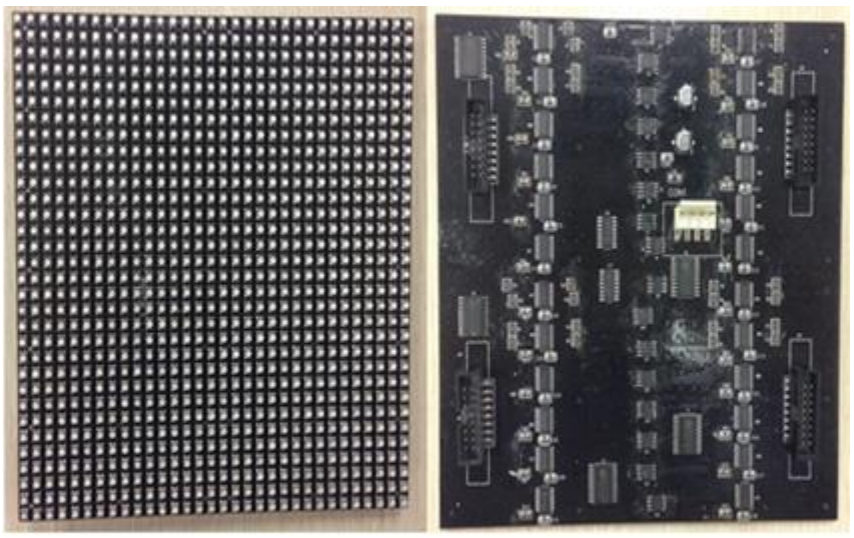


Figure 32 The front view and back view of the a single LED module consisting of 32 ×32 LED pixels, Fig. 32(a) represents the front view, and Fig. 32(b) represents the back view.

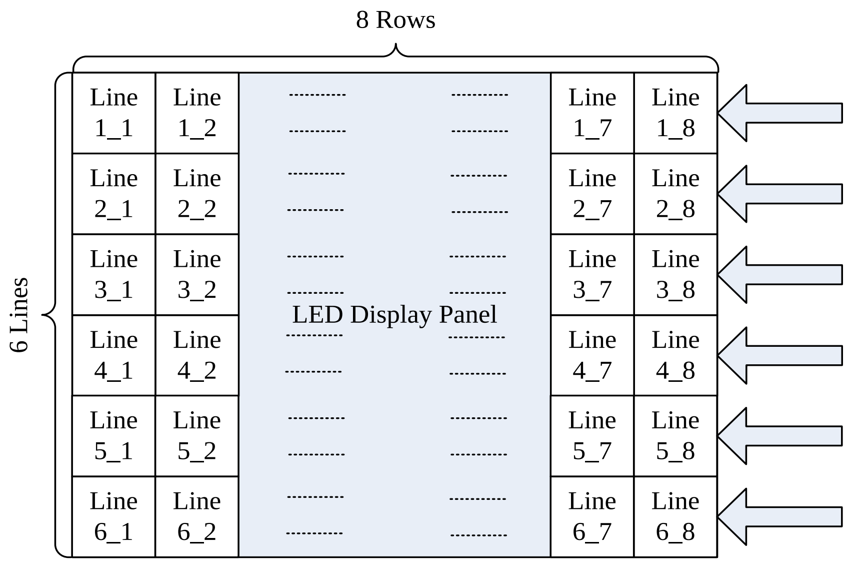


Figure 33 Energy-efficient full-color LED display system architecture.

3.3 Software implementation

3.3.1 Architecture of FPGA controller

This section presents the overall architecture of FPGA controller and the main functions of FPGA controller. FPGA controller system consists of three modules, including the data acquisition module, the bit-plane separation module, and the driving controller module.

The data acquisition module is mainly responsible for collecting and adjusting the RGB video signals and the control signals output by the HDMI decoding circuit. This module includes two sub-modules: static random-access memory (SRAM) controller module and the data image correction module. This system uses two external SRAMs to simultaneously conduct read and write operations. When one SRAM performs writing operation, the other SRAM conducts reading operation. The image data correction module will adjust and correct the image data according to the corresponding algorithm. The bit-plane separation module is responsible for converting and caching the collected video signals. The bit-plane separation module separates the 0-7 units of the RGB values, and writes them into the dual-port RAM (Random-Access Memory). The driving controller module is responsible for adjusting the luminance according to the brightness algorithm and for outputting the video signals of LED display. The driving controller module reads the data from dual-port RAM, and converts the data according to the timing required by the LED decoding circuit. The processed data is then output to LED driving circuit, driving LED display to present the intercepted video image.

The overall architecture of FPGA controller can be seen in Fig. 34. The blue line represents the data signal channel and the red line represents the clock signal channel.

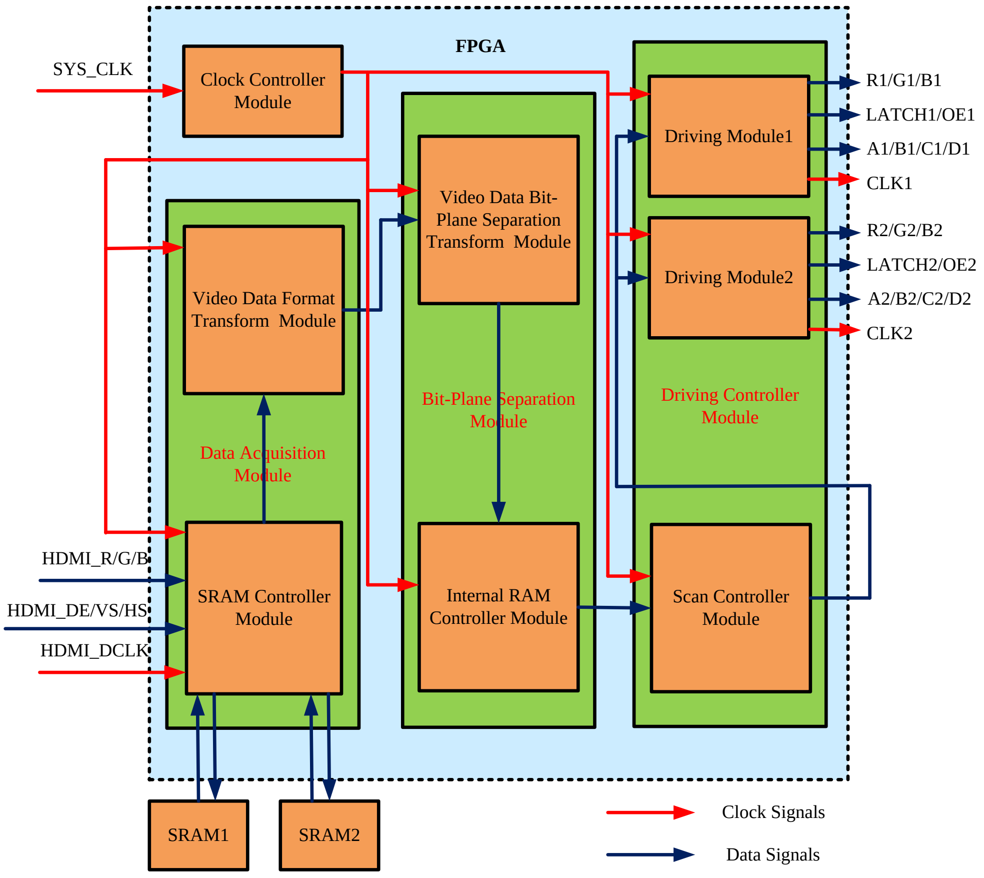


Figure 34 The architecture of FPGA controller system.

In the data signal channel, the video data source is processed by the HDMI decoding circuit, and then sent to the FPGA chip. Normally, the resolution of video signals output by HDMI is 1920 × 1200, and all the 1920 × 1200 pixels will be sent to the FPGA. Since the LED display panel can only presents 256 × 192 pixels, the video data needs to be intercepted and then stored in the external SRAM.

Then, the video data of the first frame will be stored in SARM1, and the video data of the second frame will be store in SARM2. After the data of the first frame is processed, it will be sent to the data correction module and then the SRAM1 is available to receive the third frame. This design can prevent the damage caused by the conflict between the cache of the first frame and the cache of the second frame. After the data is processed by the data correction module based on the corresponding algorithm, it will be sent to the bit-plane separation module. Each pixel of the video data consists of 8 bits R, 8 bits G and 8 bits B, and it is transmitted in a serial manner. The driving controller module will separately operate the bit 0- bit 6 and the bit 7 of the RGB values, so the bit-plane separation is necessary.

The bit-plane separation module simultaneously operates the RGB values, and stores them separately. The R value is separated and stored in its corresponding area of dual-port RAM, so is the G value and the B value. The corresponding area of R, G and B in dual-port RAM consists of 2 parts: one part is used to store the bit 0- bit 6, whereas the other part is used to store the bit 7. These two parts can conduct read and write operation at the same time. To prevent the image damage, the system employs dual-port RAMs. After the bit-plane separation module writes the first intercepted frame into the first RAM, the driving controller will display the data in the first dual-port RAM. The second intercepted frame will be written into the second RAM and then displayed. The third intercepted frame will be written into the first dual-port RAM.

Finally, the driving controller module will read the data of the bit 0 - bit 6 and the bit 7 of dual-port RAM respectively. It then converts the data in the bit 0 - bit 6 and outputs the data to the first module, and converts the data in the bit 7 and transmits the data to the second module, according to the timing requirements of the driving modules. Then the driving modules will convert the data into the RGB signals and control signals of the first interface and the data for the second interface, and then transmit them to the LED display. Finally, the video images are presented on the LED display.

In the clock signal channel, the original clock signal is provided by the crystal oscillator circuit, and it is transmitted to the PLL module in the FPGA. Then the PLL module can generate the SYSTEM_CLK clock signals which are transmitted to every module of the FPGA and serve as the working clock. In addition, for the data acquisition module, the clock which writes the data from the data acquisition module to the external SRAM comes from the HDMI decoding circuit, whereas the clock which reads the data from the SRAM for further processing comes from the SYSTEM_CLK clock.

3.3.2 Simulation results

The timing of each module has been verified via the Modelsim software, and it has met the requirements of the LED video display system. In the practical hardware system, the video data input to the FPGA controller comes from the HDMI decoding circuit. But in the simulation, this transmission process has been simplified by the introduction of HDMI timing module: dvi_sim. The structure of dvi_sim can been seen in Fig. 35.

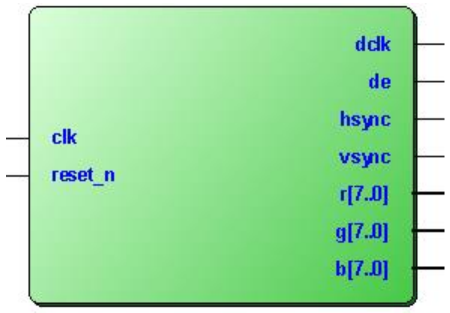


Figure 35 Structure of HDMI timing module.

By providing a system clock signal and a reset signal, the dvi_sim can output control signals (e.g., dclk/de/hsync/ysync/r/g/b signals) that have the same timings as the signals output by the HDMI decoding circuit. The video data is fixed so that the LED display presents a static picture. Fig. 36 shows the timing diagram of signals output by the dvi_sim module.

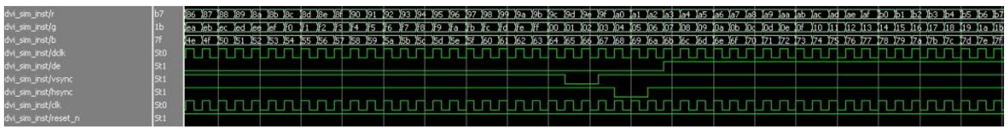


Figure 36 The simulated structure of HDMI interface module.

Fig. 37 presents the timing diagram of signals written from the bit-plane separation module into the dual-port RAM.

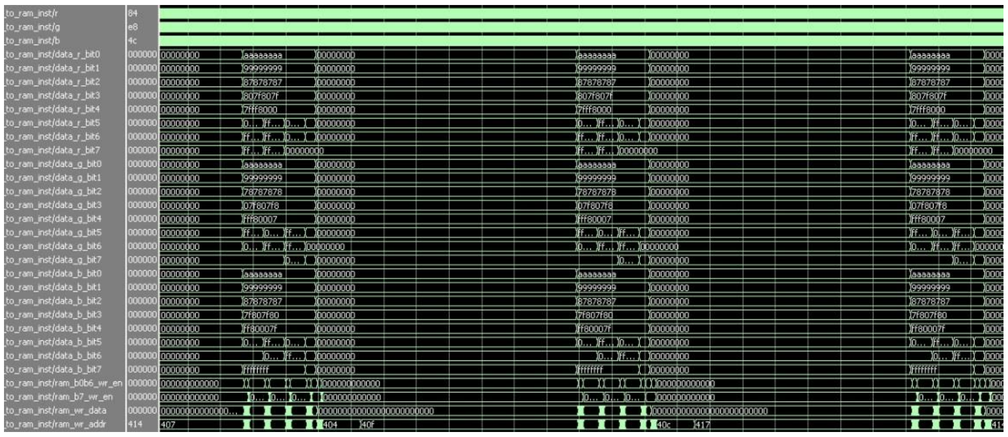


Figure 37 Simulated results of bit-plane conversion.

Fig. 38 presents the timing diagram of signals written from the bit-plane separation module into the dual-port RAM, after the signals are enlarged.

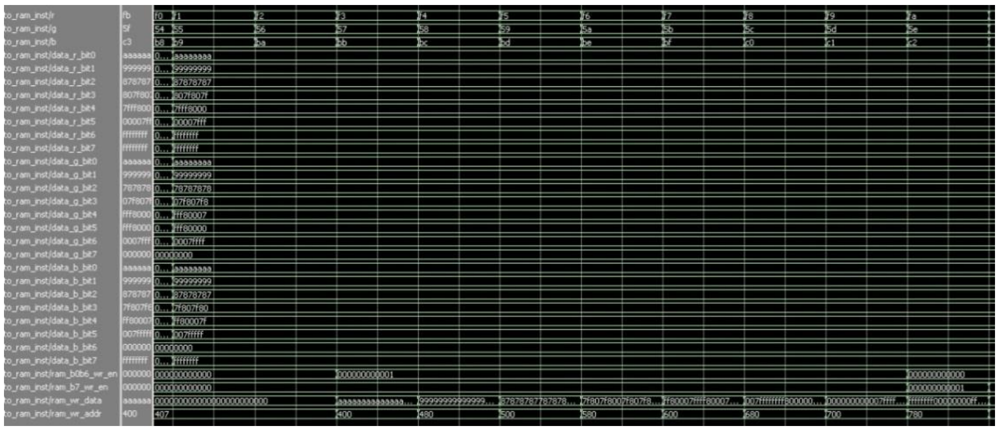


Figure 38 Details of simulated results of bit-plane.

Fig. 39 shows the timing diagram of the signals of led_sub_module. The driving controller module generates these signals and then the FPGA controller outputs them to the LED driving circuit.

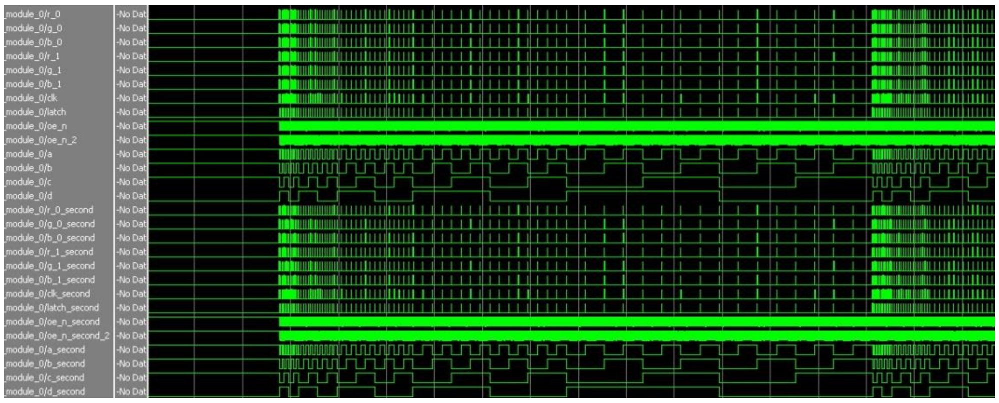


Figure 39 Simulated results of interfaces of LED display.

Fig. 40 shows the timing diagram of the signals of led_sub_module, after the signals are enlarged.

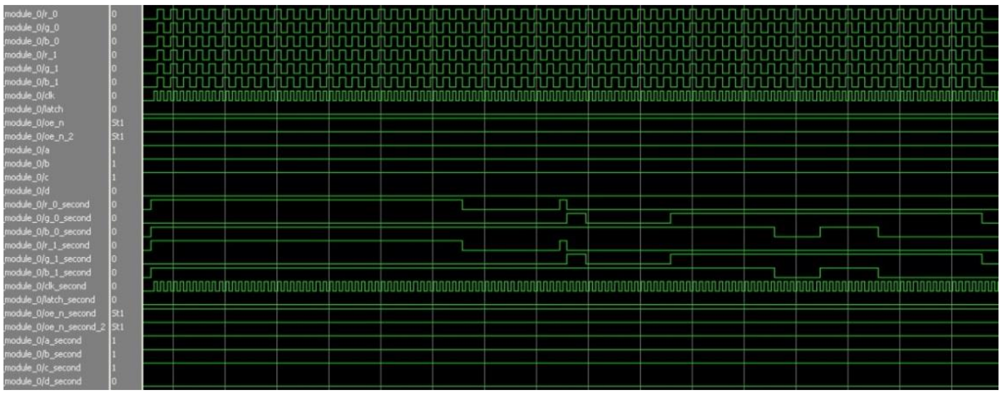


Figure 40 Details of simulated results of interfaces of LED display.

Table 4 The controller tools.

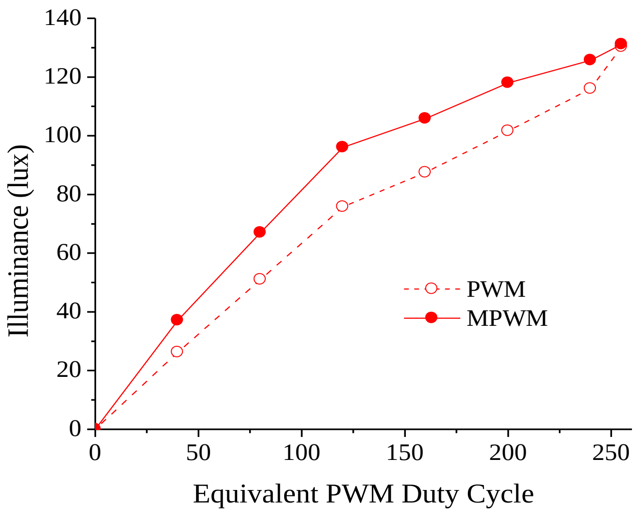
Tools	Name
FPGA	Altera Cyclone EP3C120
Computer Language	Verilog HDL
Software	Quartus II 14.1
Simulation tool	Modelsim 10.2

3.4 Experimental verification

In order to demonstrate the effectiveness of the proposed algorithm, energy-saving algorithm based on LED display panel is used. Since the full-color image is composed of a mixture of three colors (e.g., red, green and blue) and they are emitted from the LED, this experiment has tested these three colors. Then, in order to realize energy efficiency of LED display, the energy-saving algorithm has been implemented to transmit the HDMI video signals to the LED display. The experiment results have proved that the signal decoding circuit can output HDMI video signals complying with the characteristics of timing sequence. The three basic colors (e.g., red, green, and blue) have been displayed properly, and the whole system worked appropriately. Moreover, the intercepted area on the original video image can be identified by comparing the video image on the LED display and that on the original video image. The large-scale LED video display panel can be achieved by using a display panel combined by several small-sized LED panels and multiple FPGA controllers.

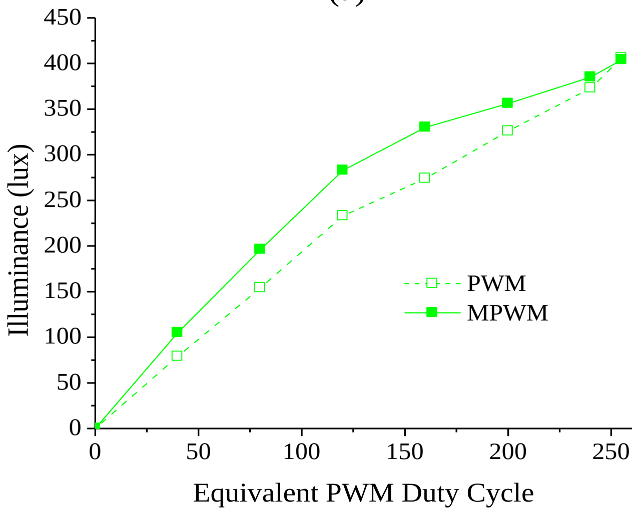
3.4.1 Experimental results on energy saving methods

An MPWM-based LED display panel system was constructed for verifying the proposed energy-saving driver design concept. A three-level driving scheme with I_1 = 10 mA and I_1 = 20 mA was adopted. PWM currents at various duty cycles (coded with 8 bits with decimal values varying from 0 to 255) were applied to an LED display module consisting of 32 $\times$ 32 LED pixels and the average luminance values were measured using a luxmeter. Similar measurements were repeated for the corresponding MPWM currents. Fig. 41(a), (b) and (c) shows the measured luminance data plotted for the red, green, and blue LED sub-pixels, respectively. It can be seen that the measured curves resemble closely the conceptualized curves shown in Fig. 18. As intended by design, the maximum luminance gain occurs in the intermediate region for all three colors. In general, the MPWM dimming method generates a higher luminance under different average current conditions . For the PWM method, the linear relationship between average current and luminance has constant slopes. For the MPWM method, it contains different piecewise linear sections with differentiated slopes.



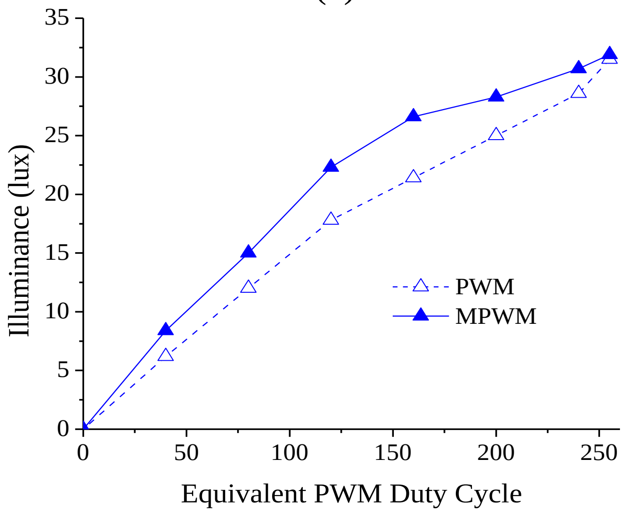
140
120
100
80
60
40
20
0
Illuminance (lux)
0
50
100
150
200
250
Equivalent PWM Duty Cycle
PWM
MPWM

(a)



450
400
350
300
250
200
150
100
50
0
Illuminance (lux)
0
50
100
150
200
250
Equivalent PWM Duty Cycle
PWM
MPWM

(b)



35
30
25
20
15
10
5
0
Illuminance (lux)
0
50
100
150
200
250
Equivalent PWM Duty Cycle
PWM
MPWM

(c)

Figure 41 Average luminance of the (a) red, (b) green, and (c) blue LED subpixels.

Fig. 45 shows the percentage gain in luminous efficacy (over PWM driving) by using the MPWM driving method, with the maximum achievable luminous efficacy gain by using DC driving included for reference. The curves shown were generated by first measuring the DC dimming curves for each color under constant junction temperatures and then calculating the areas bounded by the DC, PWM, and MPWM dimming curves and the horizontal axis. A maximum luminance gain due to MPWM dimming technique is found in the mid-range average current with 25.0%, 26.1%, and 22.2% achieved for the red, green and blue LEDs, respectively. The average luminance gain over full range average current is 10.5%, 14.1%, and 12.3% for the red, green and blue LED, respectively. In general, when the LED display system presents video images, the RGB LEDs' intensities change with the time. Thus, to achieve the most significant luminance gain, the time-averaged currents and intensities of RGB LEDs are located in the middle region.

The experiments were performed in a sealed oven at the intended junction temperature, as can be seen in Fig. 46. A pulsating DC current at 200 Hz and 1% duty cycle was used to drive the LEDs to prevent them from self-heating so that their junction temperature was kept approximately equal to the oven temperature. The peak value of the pulsating DC current was varied from 0 to 20 mA and the luminance of the LEDs were measured using lux-meter via a light guide inserted into the oven through a small opening. The same process was repeated for three junction temperatures. The luminous outputs of red, green and blue LED at different current levels have been plotted, as can be seen in Fig. 42, Fig. 43 and Fig. 44.

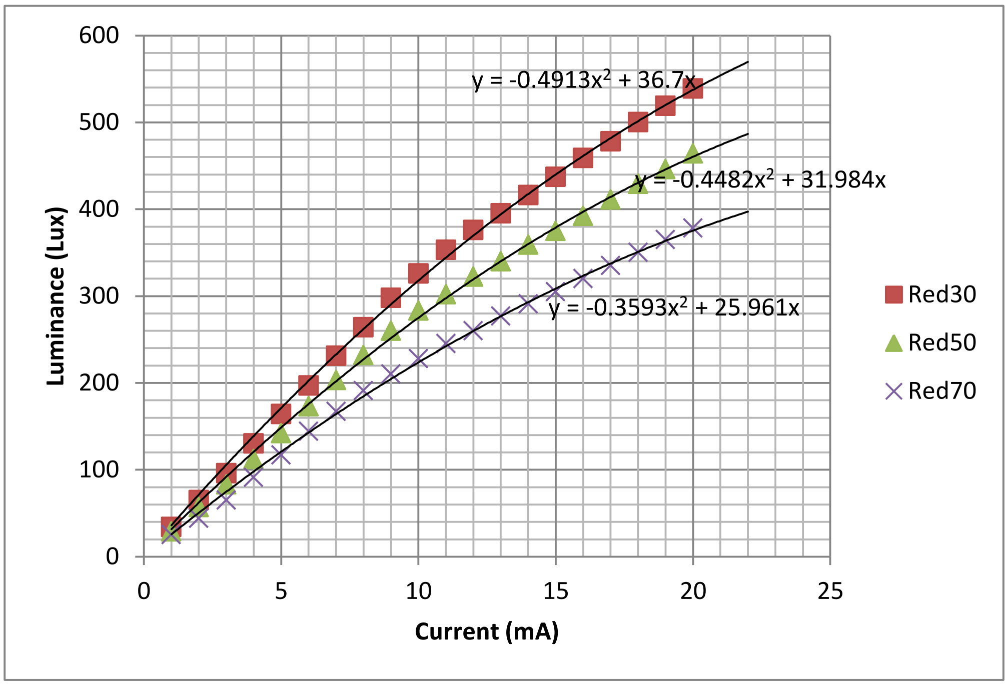


Figure 42 The luminance output of red LED.

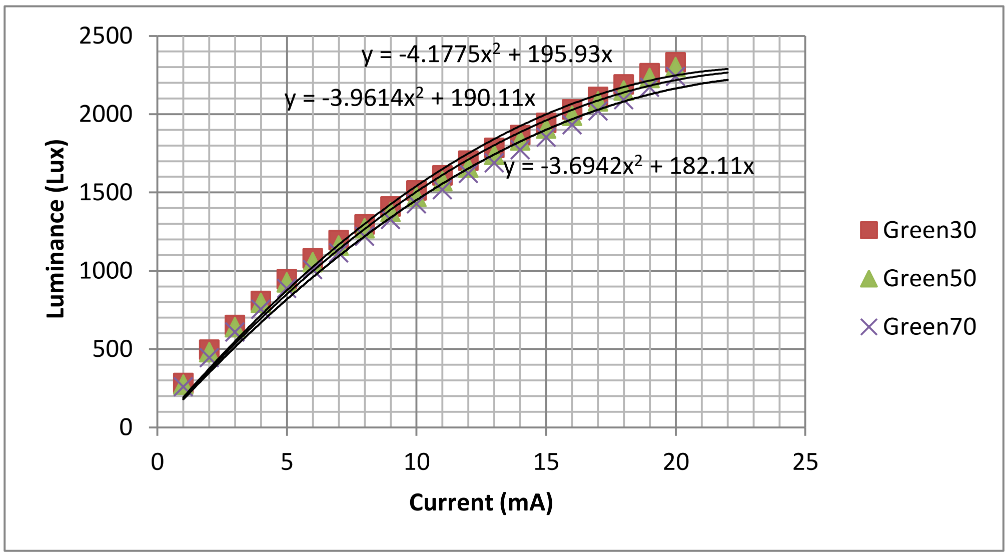


Figure 43 The luminance output of green LED.

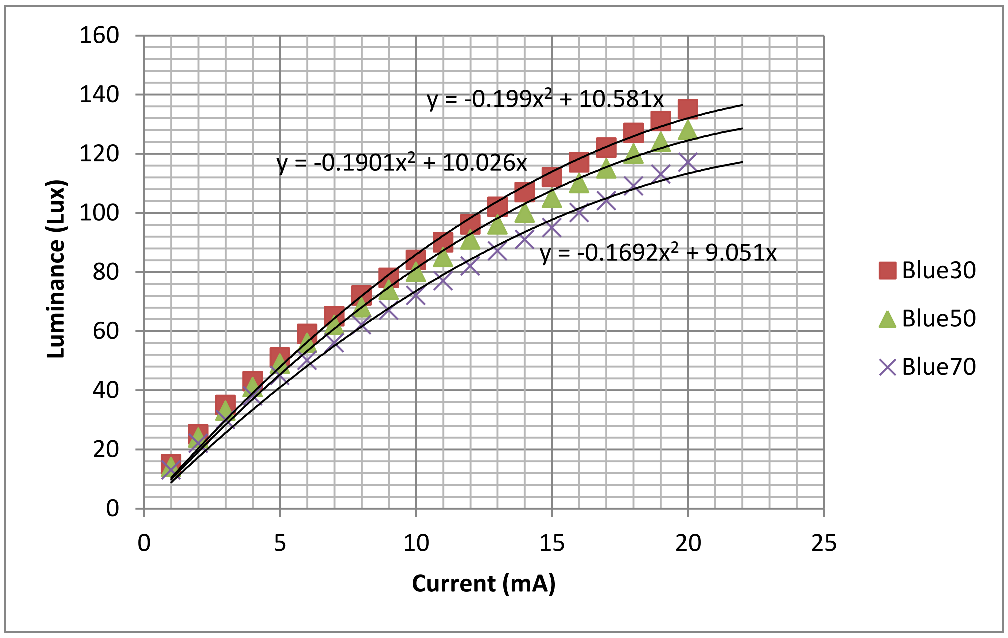


Figure 44   The luminance output of blue LED.

For each junction temperature, the measured luminance was plotted against the peak value of the pulsating DC current. The dimming curves associated with the MPWM and PWM driving methods were superimposed on the graph, and the areas bounded by them and the horizontal axis were calculated. Finally, the differences in the bounded areas were computed and plotted in Fig. 45.

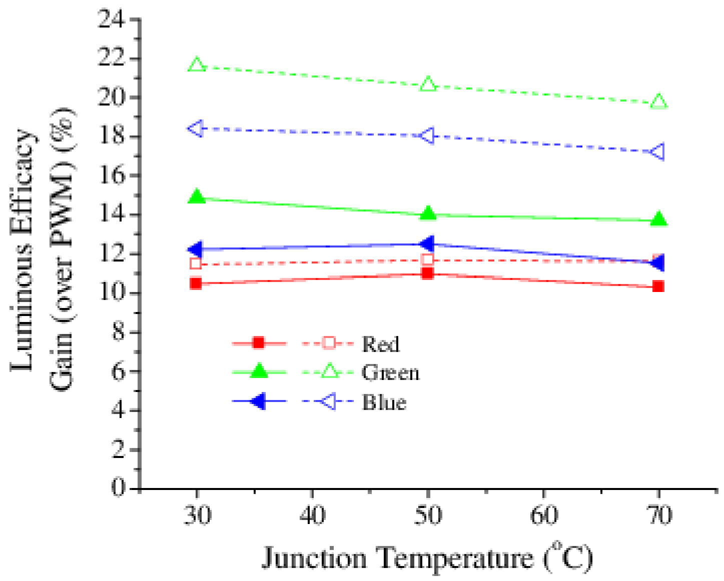


Figure 45 For each of the three colors, dashed lines indicate the maximum achievable luminous efficacy gain (by using dc driving) over PWM driving, and solid lines with marks indicate the luminous efficacy gain achieved by using MPWM driving under various junction temperatures.

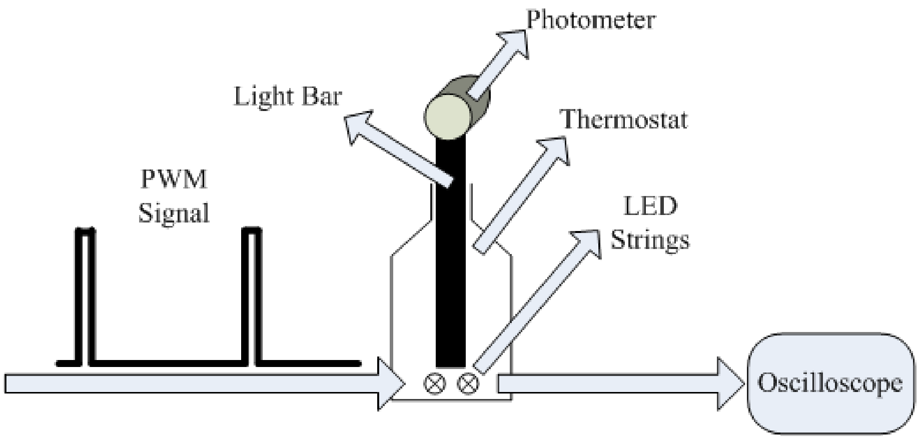


Figure 46 Setup of experiment for RGB LEDs.

Fig. 47 shows the prototype of LED display panel constructed in our laboratory, which consists of 2 × 3 LED display modules, or 64 × 96 LED pixels. The three LED display modules (showing red, green, blue color, respectively) belonging to the upper row were driven by PWM current, and those belonging to the lower row were driven by MPWM current having the same average value as the PWM current. It can be seen (when viewed at some distance from screen or color-printed on paper) that for all three colors the lower row shows a higher luminosity compared to the upper row.

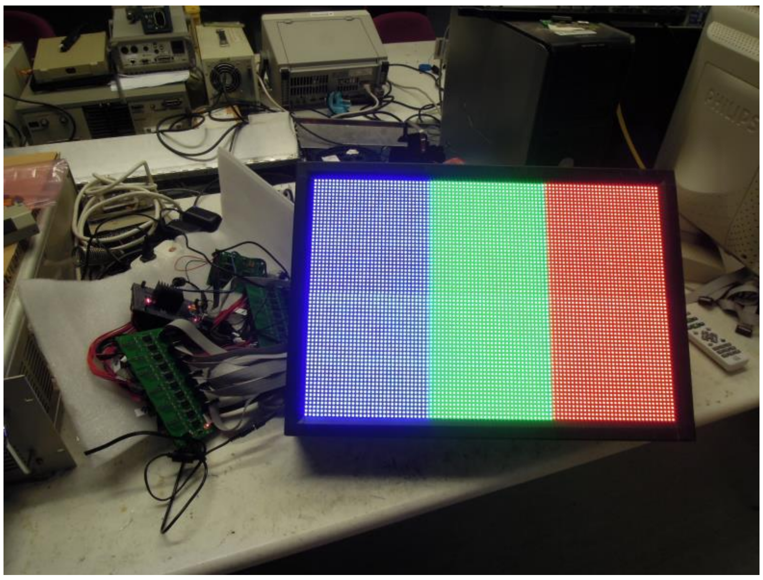


Figure 47 Prototype of LED display panel consisting of 2 × 3 LED display modules or 64 × 96 LED pixels.

3.4.2 Verification of LED display system performance

In this section, we verified the performance of the LED video display, including the chrominance uniformity, the luminance uniformity, and the functionality of the entire display system.

The chrominance uniformity needs to be measured by a specific equipment called photoelectric colorimeter. For LEDs in each primary color (red, green or blue), the wavelength of a large number of LEDs will be tested respectively. If the range of the wavelength is wide, the chrominance uniformity will be weak. To ensure that each primary color is consistent, we select the LEDs whose range of wavelength is below 5 nm to form the LEDs of each color (red, green, or blue).

The luminance uniformity is determined by the timing of the control system, the data correction, and the driving circuit. So the system functionality of the LED video display can be further verified by testing the consistency of the LED luminance. The luminance of each LED was measured by the light meter in this study. The LED panel system designed in this study has 48 modules, and each module has 1024 LEDs. To ensure the luminance uniformity of the entire system, the luminance of an area in each module was randomly tested. The luminance value of each LED pixel is determined by the pictures with solid color output by the HDMI video source unit. The performance of the luminance uniformity with solid colors can be seen in Fig. 48.

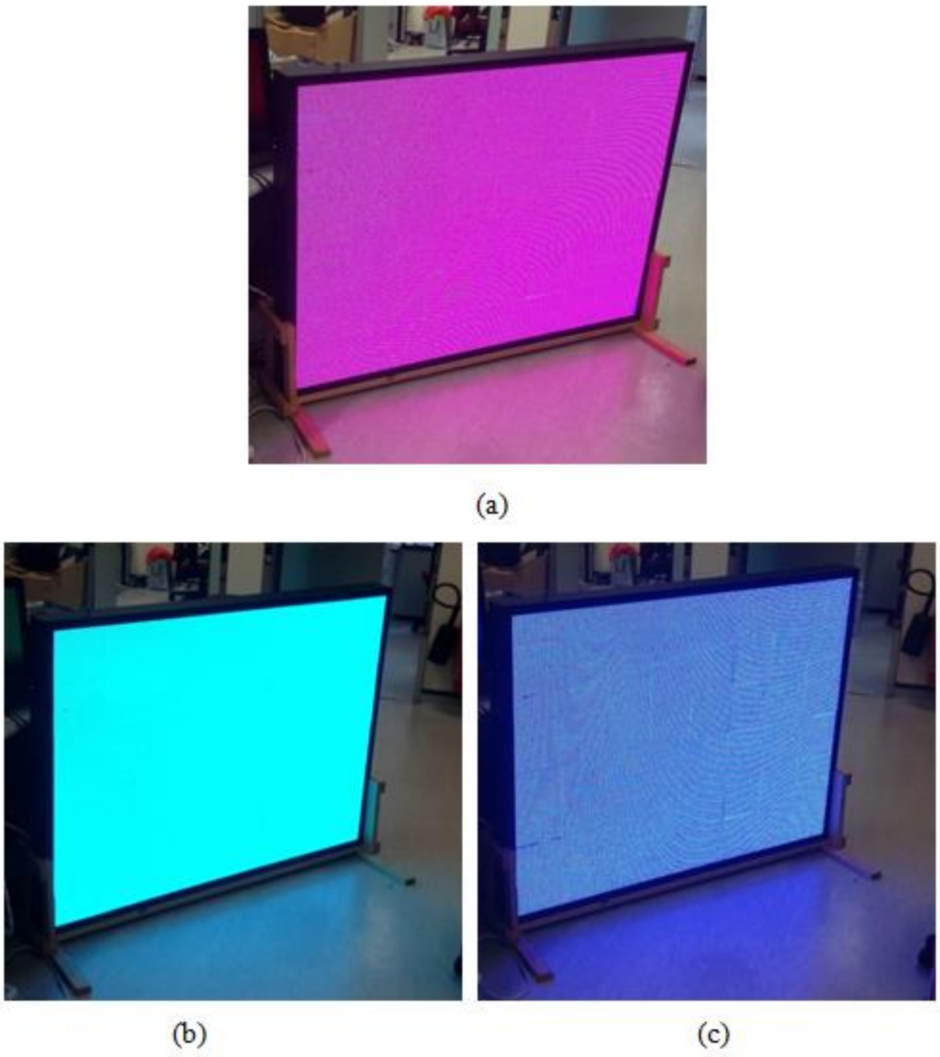


Figure 48 The luminance uniformity test with solid colors.

The luminance difference among each LED can be calculated using the following formula:

$$L_{\text{difference}} = \frac{\left|L_{\text{i}} - L_{\text{ave}}\right|}{L_{\text{ave}}} \times 100\% \qquad (3\text{-}4)$$

$L_{\text{difference}}$: the difference between original brightness value and the average brightness value

L_i : the luminance value for each of LED module

L_{ave} : the average luminance value of all LED modules

$$L_{ave} = \frac{1}{n}\sum_{i=1}^{n} L_i \qquad (3\text{-}5)$$

where L_{ave} is the average of N measured LED modules, N = 48 in this case

The difference between original luminance value and the average luminance in standard module has been tested, based on the above equations. The testing results of Fig.48 (a), (b), and (c) are illustrated in Fig. 49, 50 and 51 respectively. The luminance differences between each of the modules for the color shown in Fig.48(a) is below 1.3%, as can be see in Fig. 49; the luminance differences between each of the modules for the color shown in Fig. 48(b) is below 0.4%, as can be see in Fig. 50; The luminance differences between each of the modules for the color shown in Fig. 48(c) is below 3.8%, as can be see in Fig. 51. These results prove the luminance uniformity of the designed LED video display system.

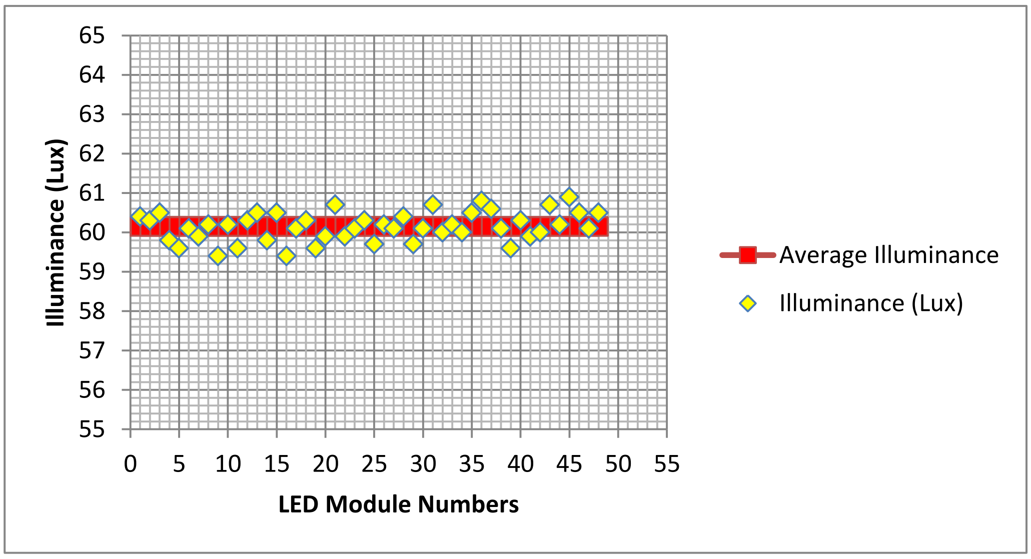


Figure 49 The luminance differences between each of the modules for the color as shown in Fig.45(a).

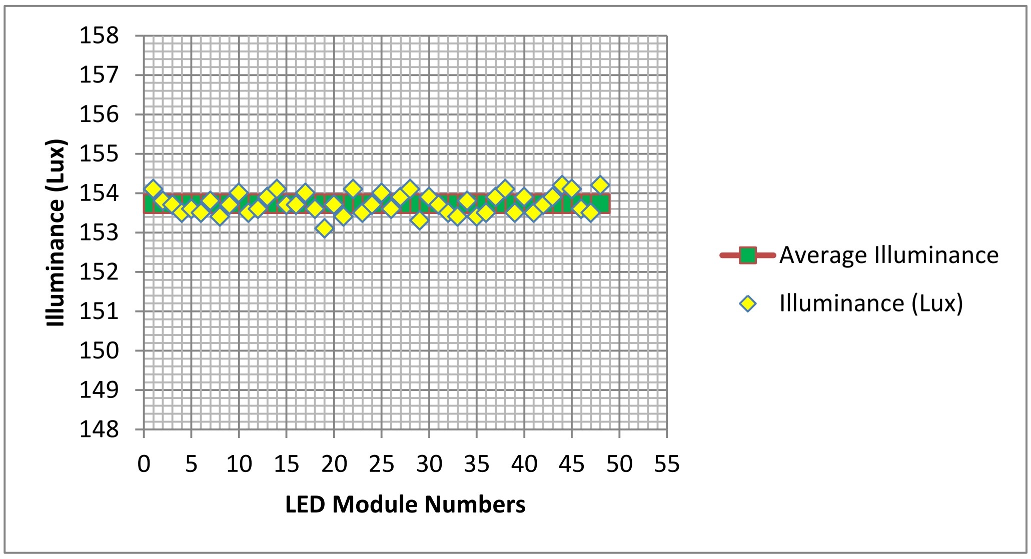


Figure 50 The luminance differences between each of the modules for the color as shown in Fig.45(b).

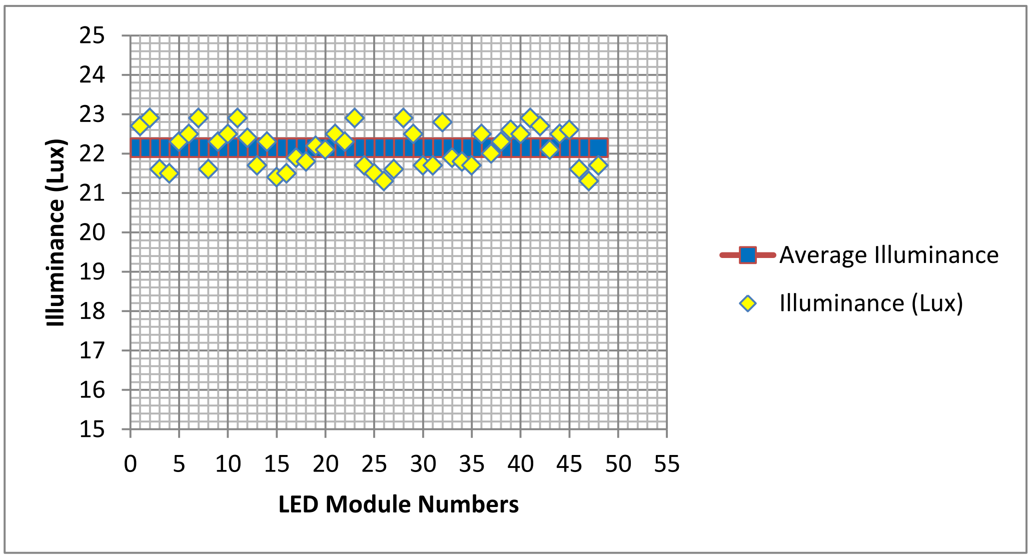


Figure 51 The luminance differences between each of the modules for the color as shown in Fig.45(c).

After the hardware and software design assembly, we verified the functionality of the entire system. It is found that the energy-saving LED video display panel and the HDMI signal decoding circuit work in a proper manner. The graphics card used in this study was the smart player. The display mode was adjusted to the “same mode on two displays”, and the PC monitor serves as the primary display through the frequency divider of the HMDI interface. The same video source files were simultaneously transmitted to the LCD display and the LED display. Since each LED module has limited pixels to display, we selected a particular part of original image to be assigned to a particular LED module. So if the video data were changed in the main display, the intercepted video data in an area of 256 × 192 could also be altered in the LED display. The LED display presents video sources (e.g., the house in the video clip) presented the original file with good performance, as can be seen in Fig. 52. The luminance of the LED display is higher than that of the LCD display.

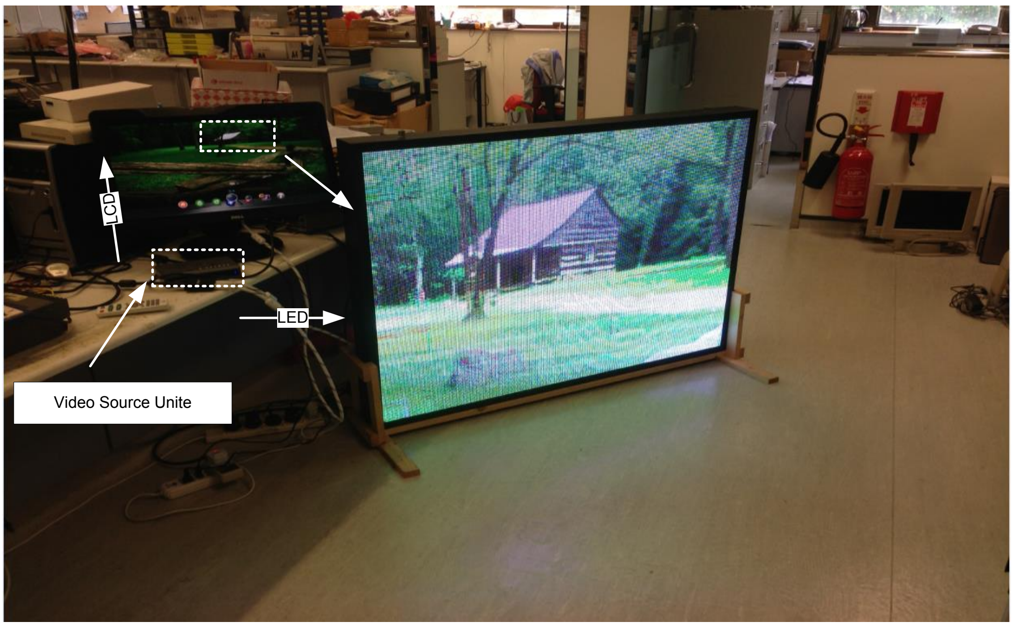


Figure 52 LED video display with good performance

The LED and the LCD video displays presented the video sources simultaneously and in real-time manner, as can been seen in Fig. 53. The location of the intercepted video images can be seen in the LCD display. The intercepted video images (e.g., the flags from various countries) have been well displayed on the LED screen.

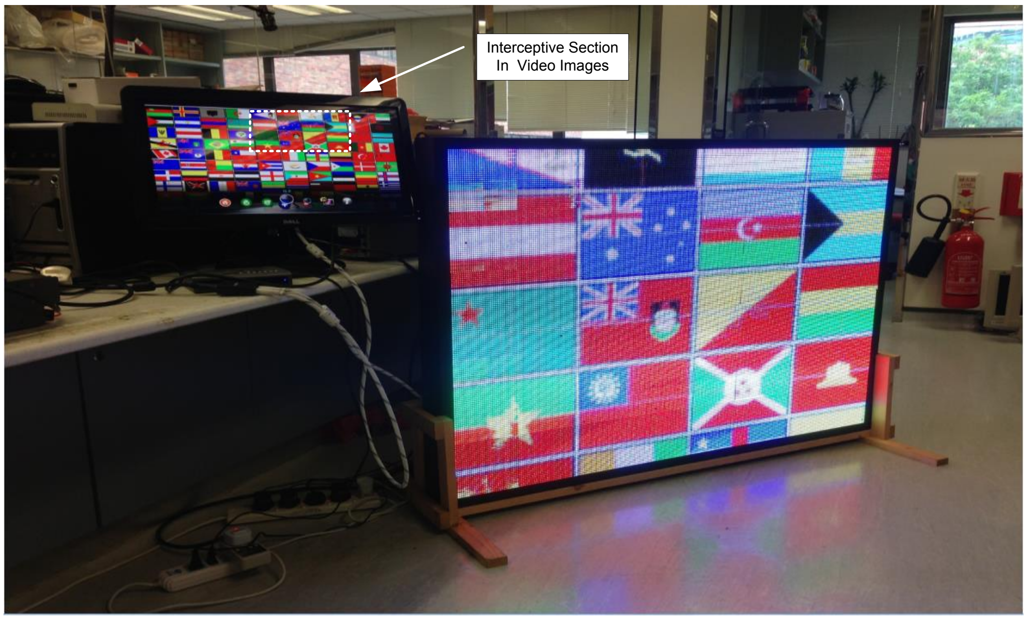


Figure 53 LCD and LED video display of flags in real-time.

Finally the setup of the LED video display system and its main components can be seen in Fig. 54. And the resolution of the LED video display is 256 × 192, and the dot pitch of the LED display is 5 mm × 5 mm. The wavelengths of the red, green and blue LEDs are 625 nm, 520 nm, 470 nm, respectively. If the frequency of the video scan and the frame rotation speed of the LED video display are high enough to be indistinguishable by human eyes, then the display appears to be constantly illuminated. The specifications of the LED video display are lists in table 5.

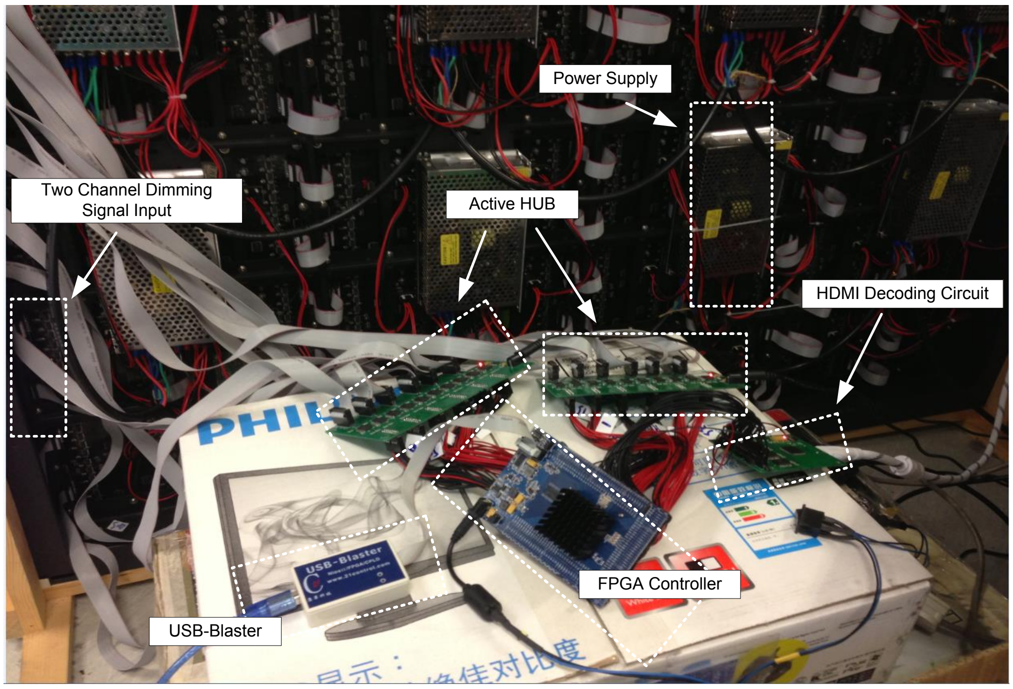


Figure 54 Setup of MPWM based LED video display system.

Table 5 Specification of the full-color LED video display system.

Item	Full-color LED display system
Pixel pitch	5 mm × 5 mm
Working voltage	AC220 V
Power consumption	Average 400 W/sqm
Pixel configuration	Tricolor SMD (1R1G1B)
Pixel size	4 mm (circle)
Pixel density	40,000 Pixels/m^2
Color spectrum	R = 625 nm, G = 520 nm, B = 470 nm
Refresh frequency	60 frames/second
Driver mode	1/16 Scanning
Grade level	RGB 256 grade/each
Viewing angle	Horizontally > 160° / Vertically > 120°
Driving IC	MBI5026
Change frame frequency	60 Hz
Input signal	DVI/HDMI
LED module size	256 pixels × 192 pixels (8" × 6")

3.5 Realization of large-area LED display system

The merits of the MPWM dimming approach have been discussed and verified in the previous section and the video images have been presented in the LED display panel with good quality. In this section, a discussion on how to expand the size of the energy-saving LED display system to realize the large-scale LED display system will be discussed.

Large-scale LED display system has the advantage of seamless connectivity, modular maintenance, natural color, even display, wide color gamut. In order to achieve large-scale LED display system, more LED display modules need to be cascaded together to form large-scale LED display system. The image consistency should be guaranteed and the number of control signals should be reduced to achieve more convenience. So the control signals, excluding the data signals, should be connected together. This design may ensure the timing consistency of the large-scale LED display system. At the same time, large-scale LED display system has the problem of color distortion at low gray level. Green and blue LEDs are made from nitride-based semiconductors which suffer more significantly from the efficiency droop effect, compared to read LEDs that are made from arsenide- or phosphide-based semiconductors. This issue can be alleviated by increasing the coordination between the driving IC and the control system of LED display. And this issue can be further reduced by adjusting the parameters of the control system of the large-scale LED display system.

In addition, the modular design of the display module needs to facilitate the expansion of LED display rows and columns. Fig. 55 shows a schematic diagram of the ranks of the expansion. The OE, CLK, and LTD signals from MBI5026 driver and A, B, C, and D signals from 74HC138D Decoder are connected in a parallel way. To extend a line, a DATA signal cable should be added. The more the number of display lines, the more DATA signal cables. Thus the width of the display is restricted by the controller output pin. Extension of columns can be obtained through direct MBI5026 way, with no additional signal control lines needed. If the number of columns increases, more time is needed to achieve LED status latch. To ensure the refresh frequency and gradation levels, a FPGA controller is used to distribute data. Multiple FPGA controllers are responsible for controlling the corresponding region of LED display, which will be further discussed in Chapter 4.

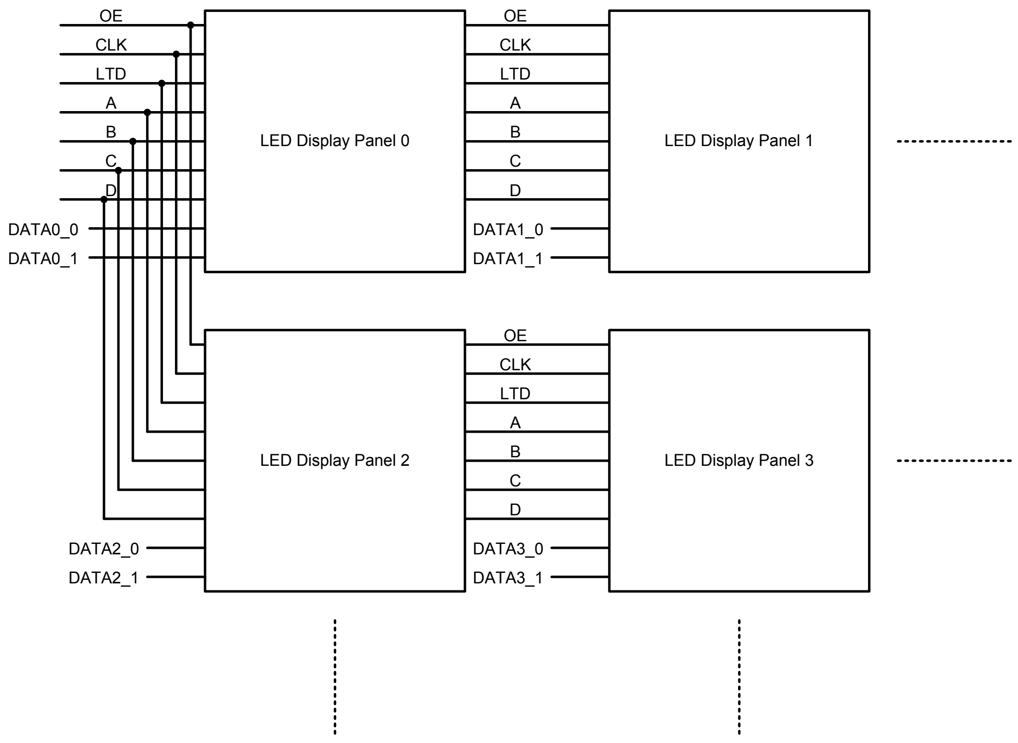


Figure 55 Large-scale LED display system consisting of standard LED display panels.

To fully display the original video source, several LED display panels can work together and each of them displays an area of the original video source.

Firstly, one of the FPGA controllers can be selected as the primary controller. This primary controller will receive the original HDMI video signals, and then deliver each part of the original video source to its corresponding FPGA controller. The other FPGA controllers will process these video signals and send them to the corresponding LED display panels. Then each LED panel can display its corresponding video image, and the original video source can be fully displayed, as can be seen in Fig. 56.

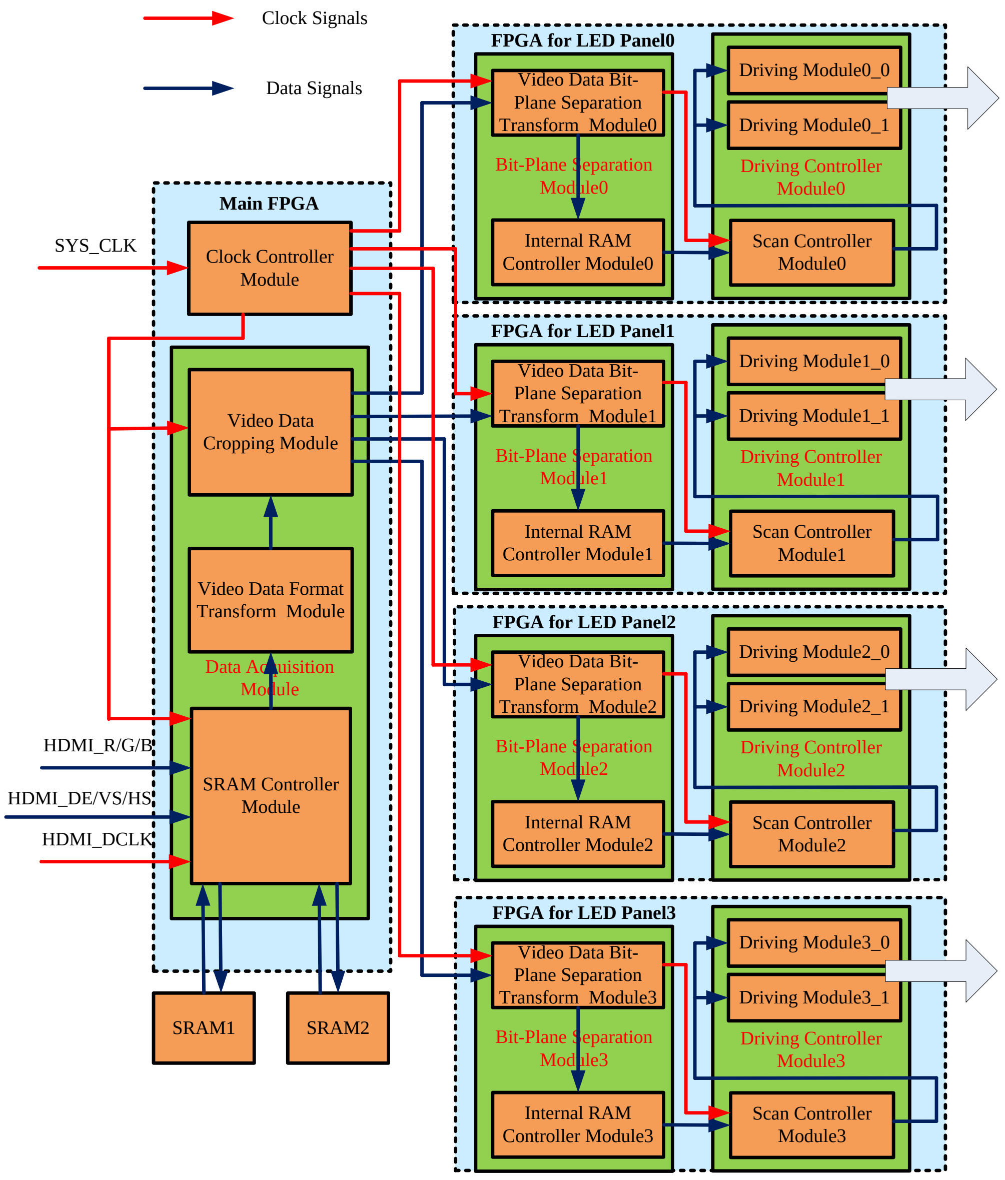

Clock Signals
Data Signals
Main FPGA
SYS_CLK
Clock Controller Module
Video Data Cropping Module
Video Data Format Transform Module
Data Acquisition Module
SRAM Controller Module
HDMI_R/G/B
HDMI_DE/VS/HS
HDMI_DCLK
SRAM1
SRAM2
FPGA for LED Panel0
Video Data Bit-Plane Separation Transform Module0
Bit-Plane Separation Module0
Internal RAM Controller Module0
Driving Module0_0
Driving Module0_1
Driving Controller Module0
Scan Controller Module0
FPGA for LED Panel1
Video Data Bit-Plane Separation Transform Module1
Bit-Plane Separation Module1
Internal RAM Controller Module1
Driving Module1_0
Driving Module1_1
Driving Controller Module1
Scan Controller Module1
FPGA for LED Panel2
Video Data Bit-Plane Separation Transform Module2
Bit-Plane Separation Module2
Internal RAM Controller Module2
Driving Module2_0
Driving Module2_1
Driving Controller Module2
Scan Controller Module2
FPGA for LED Panel3
Video Data Bit-Plane Separation Transform Module3
Bit-Plane Separation Module3
Internal RAM Controller Module3
Driving Module3_0
Driving Module3_1
Driving Controller Module3
Scan Controller Module3

Figure 56 The software realization of large-scale LED display system.

Based on the above analysis, the large-scale LED video display system can be obtained via the hardware and software realization. At the same time, since the large-area LED display system contains more LED display panels, a larger amount of the power consumption can be saved through the energy-saving methods.

3.6 Summary

In this chapter, the modified LED video display system architecture has been firstly proposed, with the detailed information on the modified hardware and software. The hardware implementation consists of three components, the HDMI decoding circuit, the FPGA controller circuit and the LED display driving circuit. For the limitation of the size of LED display panel in the lab, part of the video data has been intercepted and displayed in the LED display panel. The energy saving dimming method has been applied in the hardware assembly of the proposed LED display panel system. In the software implementation, the architecture of FPGA controller has been provided in regard to the data signal channel and the clock signal channel. There are three components in the FPGA controller. The first component is the data acquisition module. It is mainly responsible for collecting and adjusting the RGB video signals and the control signals output by the HDMI decoding circuit. After the signals are transferred to the bit plane separation module, the second component of the FPGA controller, they will be converted and cached in the internal RAM. Then all the data signals and control signals will be output into the driving controller module, the third component of the FPGA controller, which is responsible for adjusting the luminance and for outputting the video signals of LED display panel. The HDMI simulation components have been added to further verify the timing correction in the software implementation, and the simulation result has been given. Finally, the experimental results on energy saving methods have been be presented, the color uniform has been tested by the light meter, and the LED display system performance has been verified. Finally, both the hardware realization and the software realization for the large-area LED display system has been proposed.

Chapter 4 Conclusion

In this chapter, we first make a conclusion about the thesis and then present the contributions of this study. Some possible future work and potential research areas for further development will also be presented.

4.1 Contribution of this thesis

In this thesis, the characteristics of VGA, DVI and HDMI video resources and the principle of CRT, LCD, and LED video displays have been discussed. Compared with traditional video displays, LED video display has more advantages in outdoor and large-scale applications. The luminous efficiency of LED video display deserves further exploration in order to improve the overall efficiency of the system. Conventional dimming methods for LED video display, including Analog or direct current (DC) dimming method, ON-OFF pulse-width modulation (PWM) dimming method, and bi-level PWM (BPWM) dimming method, have been reviewed in detail. The differences in luminous intensity and efficacy between the AM and PWM driven LEDs have been analyzed and discussed. It is found that the luminous efficacy under AM dimming methods is always higher than that of PWM dimming methods. In some cases, the efficacy difference between the two conventional dimming methods can be as much as 30%. The advantages and disadvantages of some conventional dimming methods (e.g., binary PWM, gate PWM, pulse amplitude modulation) have also been discussed from the perspective of flexibility of luminous control.

Then the basics of LED video display, the system architecture, the proposed multi-level dimming methods, and the algorithm of the video data correction have also been provided. The LED video display system consists of data source transmission module, signal processing module, and LED display module, and these modules coordinate to display the video image on the display panel. The luminous efficacy of LED video display panel can be improved by introducing a DC-offset into the conventional PWM dimming current. For a three-level driving scheme, the criterion for selecting the mid-level current in order to maximize the luminous efficacy gain has been derived mathematically. In utilizing the multi-level current driving technique, the dimming flexibility of LED can be retained by using duty cycles as a control parameter for adjusting the luminous intensity. To solve the nonlinearity introduced by MPWM driving method and preserve color quality, the average MPWM current has been estimated to produce the same luminance as the original PWM current. The algorithm of anti-gamma correction has been discussed to correct the video signals used in CRT display so that they can also be well employed by the LED display.

Finally, a FPGA-based LED video display panel system was constructed for verifying the proposed energy-saving driver design concept. The design and implementation details of the multi-level current dimming have been discussed. In the hardware implementation, the existing LED video display configuration can be extended at low cost for the realization of the MPWM driving method by the use of multiple LED driver modules and simple video signal conversion. In the software implementation, the architecture of FPGA controller has been given in regard to the data signal channel and the clock signal channel. Then the simulation results of the FPGA controller have been provided to verify the timing and the data format. Then the average luminous efficacy gain over the conventional PWM-based system was measured for verifying the effectiveness of the proposed MPWM-based system. The proposed adaptive multi-level current driving technique has been implemented and its effectiveness has been successfully demonstrated on LED video display panels through experiments. The three colors of RGB LED display can be well displayed, and the intercepted video images have been presented correctly in the LED screen.

The first contribution of this thesis is the proposal of the MPWM driving method, which can result in a significant energy saving in large-area LED display systems. A three-level driving scheme is suggested as a practical choice for achieving a compromise between dimming linearity and luminous efficacy. It is showed that the luminous efficacy by using the MPWM driving method can be enhanced with the maximum achievable luminous efficacy gain by using DC driving method.

The second contribution of this thesis is the design and implementation of modified LED video system, namely, the MPWM-based large-area LED display system. The advantages of this design and implementation include: 1) The data splitter and data correction can be implemented on the FPGA without incurring extra cost compared to existing systems; 2) although the number of LED driver modules is doubled in the modified system, the overall cost is not doubled, since IC manufacturers can modify their ICs to include two or more such modules within the same IC package.

In general, the size of a full-color large-area LED display panel for outdoor application is 100 square meters. The average power consumption for this type of LED display is 500 W/sqr. Assume that the LED display is lighted for 20 hours per day. The total power consumption of this type of LED display is 365,000 kWh in a year. As discussed in the experimental verification, the maximum luminance gain for RGB LEDs is at least 22.2 % by using the proposed energy-saving MPWM dimming method. Therefore, the energy consumption reduced by our proposed LED display is 81,030 kWh annually. The cost of one kWh is 0.16 US dollar, and thus the total cost saved by the proposed LED display is 12,964 US dollar annually. The saved cost for a period of five years is equivalent to the purchasing price of a new LED display.

4.2 Suggestions for future work

The experimental verification of the proposed full-color LED video display system was performed on the LED display panels which were actually modified from existing commercial products with predesigned architecture. For these products, the same supply voltage of 5 V is used to power the LEDs and other onboard ICs. Considering that the forward voltage of the red, green, and blue LED is 2.0 V, 3.2 V, and 3.2 V, respectively, at the desired forward current, the minimum supply voltage required for driving the LEDs while maintaining linear operation of the constant-current drivers is about 3.2 + 1 = 4.2 V. Lowering the supply voltage to (maximum LED forward voltage + 1) V will improve the driver efficiency while maintaining linear operation of the constant-current drivers. This gives better driver efficiency if other onboard ICs can operate normally under the lowered supply voltage. This option may be considered when the system is optimized for commercialization.

To further increase the size of the LED display panel, the data management algorithms for improving the data processing efficiency of the FPGA controller may be studied in order to cope with the increased video data volume associated with the expansion of the panel size. In some cases, there is a long distance between the center of control system and the LED display panel, so the hardware and software implementation of wireless data transmission and some related technical issues may be further explored. The hardware and software implementation of three-dimensional (3D) viewing effect which strengthens the illusion of depth perception can be considered for the LED video display panel system. As 3D video sources became increasingly successful and popular throughout the recent decade, the corresponding design for the 3D video LED display deserves further investigation.

Appendix A

Details of the module input and output pins are as below in table 6.

Table 6 Description of the input and output of FPGA controller system.

Number	Port	Location	I/O	Description
1	dvi_de	W20	I	Output data enable
2	dvi_vsync	AB21	I	Vertical Sync signal
3	dvi_hsync	AD22	I	Horizontal Sync signal
4	dvi_dclk	AF24	I	Clock signal
5	dvi_r[0]	Y18	I	Red color signal from HDMI
6	dvi_r[1]	AA18	I	Red color signal from HDMI
7	dvi_r[2]	W19	I	Red color signal from HDMI
8	dvi_r[3]	Y19	I	Red color signal from HDMI
9	dvi_r[4]	Y21	I	Red color signal from HDMI
10	dvi_r[5]	Y20	I	Red color signal from HDMI
11	dvi_r[6]	W24	I	Red color signal from HDMI
12	dvi_r[7]	W23	I	Red color signal from HDMI
13	dvi_g[0]	Y23	I	Green color signal from HDMI
14	dvi_g[1]	Y22	I	Green color signal from HDMI
15	dvi_g[2]	AA20	I	Green color signal from HDMI
16	dvi_g[3]	AA19	I	Green color signal from HDMI
17	dvi_g[4]	AA17	I	Green color signal from HDMI
18	dvi_g[5]	Y17	I	Green color signal from HDMI
19	dvi_g[6]	AC20	I	Green color signal from HDMI
20	dvi_g[7]	AB20	I	Green color signal from HDMI
21	dvi_b[0]	AD21	I	Blue color signal from HDMI
22	dvi_b[1]	AE21	I	Blue color signal from HDMI
23	dvi_b[2]	AD20	I	Blue color signal from HDMI
24	dvi_b[3]	AE20	I	Blue color signal from HDMI
25	dvi_b[4]	AC19	I	Blue color signal from HDMI
26	dvi_b[5]	AD19	I	Blue color signal from HDMI
27	dvi_b[6]	AB18	I	Blue color signal from HDMI
28	dvi_b[7]	AC18	I	Blue color signal from HDMI
29	reset	A6	I	System reset
30	sys_clk_in	AE14	I	System clock
31	r_serial[0]	AA24	O	Red color signal (channel0)

32	g_serial[0]	V20	O	Green color signal (channel0)
33	b_serial[0]	AC25	O	Blue color signal (channel0)
34	r_serial[1]	AC24	O	Red color signal (channel0)
35	g_serial[1]	W25	O	Green color signal (channel0)
36	b_serial[1]	AB24	O	Blue color signal (channel0)
37	a[0]	Y24	O	Row selection (channel0)
38	b[0]	AB23	O	Row selection (channel0)
39	c[0]	W26	O	Row selection (channel0)
40	d[0]	Y26	O	Row selection (channel0)
41	clk[0]	Y25	O	Clock signal for display (channel0)
42	latch[0]	AA26	O	Latch signal for display (channel0)
43	oe_n[0]	AA23	O	Enable signal for display (channel0)
44	a[1]	AC22	O	Row selection (channel1)
45	b[1]	V22	O	Row selection (channel1)
46	c[1]	V21	O	Row selection (channel1)
47	d[1]	W22	O	Row selection (channel1)
48	clk[1]	AD25	O	Clock signal for display (channel1)
49	latch[1]	AB22	O	Latch signal for display (channel1)
50	oe_n[1]	W21	O	Enable signal for display (channel1)
51	r_serial_s[0]	AB26	O	Red color signal (channel1)
52	g_serial_s[0]	AC23	O	Green color signal (channel1)
53	b_serial_s[0]	AB25	O	Blue color signal (channel1)
54	r_serial_s[1]	AD23	O	Red color signal (channel1)
55	g_serial_s[1]	AC26	O	Green color signal (channel1)
56	b_serial_s[1]	AD26	O	Blue color signal (channel1)

Appendix B

The FPGA codes are listed in this section.

1) Top module FPGA codes

```
`include		"../source/define.v"


module led
(
	dvi_r,
	dvi_g,
	dvi_b,
	dvi_dclk,
	dvi_de,
	dvi_vsync,
	dvi_hsync,

	clk,
	latch,
	oe_n,

	a,
	b,
	c,
	d,

	r_serial,
	g_serial,
	b_serial,
	r_serial_s,
	g_serial_s,
	b_serial_s,

	clk_s,
	latch_s,
	oe_n_s,

	a_s,
	b_s,
	c_s,
	d_s,

	led_disp,

	system_clk_in,
	reset
);


input	[7:0]	dvi_r;
input	[7:0]	dvi_g;
input	[7:0]	dvi_b;
input			dvi_dclk;
input			dvi_de;
input			dvi_vsync;
input			dvi_hsync;

output	[11:0]	r_serial;
output	[11:0]	g_serial;
output	[11:0]	b_serial;

output	[11:0]	r_serial_s;
output	[11:0]	g_serial_s;
output	[11:0]	b_serial_s;

output	[5:0]	clk;
output	[5:0]	latch;
output	[5:0]	oe_n;

output	[5:0]	a;
```

```
output	[5:0]	b;
output	[5:0]	c;
output	[5:0]	d;

output	[5:0]	clk_s;
output	[5:0]	latch_s;
output	[5:0]	oe_n_s;

output	[5:0]	a_s;
output	[5:0]	b_s;
output	[5:0]	c_s;
output	[5:0]	d_s;

output	[3:0]	led_disp;

input			system_clk_in;
input			reset;


wire			reset_n;
wire			reset_in_n;

reg		[25:0]	system_clk_in_cnt;
wire			system_clk_in_g;

reg		[25:0]	system_clk_cnt;
wire			system_clk;
wire			sys_clk_locked;

wire	[11:0]	ram_b0b6_wr_en;
wire	[11:0]	ram_b7_wr_en;
wire	[95:0]	ram_wr_data;
wire	[10:0]	ram_wr_addr;

wire			system_clk_2x;

wire			disp_req;
wire			disp_ack;
wire			disp_buff_sel;

wire	[255:0]	cs_data_module_0;
reg		[31:0]	cnt_vsync;

/*			main
									*/

`ifdef		__ALTERA__

assign		reset_in_n		= reset;
assign		reset_n			= reset & sys_clk_locked;

assign		led_disp[0]		= system_clk_in_cnt[25];
assign		led_disp[1]		= system_clk_cnt[25];
assign		led_disp[3:2]	= cnt_vsync[31:30];

always @ (posedge system_clk_in or negedge reset_in_n)
begin
	if (~reset_in_n)
		system_clk_in_cnt	<= 26'h0;
	else
		system_clk_in_cnt	<= system_clk_in_cnt + 26'h1;
end

always @ (posedge system_clk or negedge reset_n)
begin
	if (~reset_n)
		system_clk_cnt		<= 26'h0;
	else
		system_clk_cnt		<= system_clk_cnt + 26'h1;
end

always @ (posedge system_clk_2x or negedge reset_n)
begin
	if (~reset_n)
		cnt_vsync				<= 32'h0;
	else if (~dvi_vsync)
		cnt_vsync				<= 32'h0;
	else
		cnt_vsync				<= cnt_vsync + 32'h1;
end

system_pll			system_pll_inst
(
		.areset		(~reset_in_n),
		.inclk0		(system_clk_in),
		.c0				(system_clk_in_g),
			.c1				(system_clk),
			.c2				(system_clk_2x),
			.locked			(sys_clk_locked)
);

`else

assign		reset_in_n		= ~reset;
assign		reset_n			= (~reset) & sys_clk_locked;

assign		led_disp[0]		= system_clk_in_cnt[25];
assign		led_disp[1]		= system_clk_cnt[25];
assign		led_disp[2]		= system_clk_in_cnt[23];
assign		led_disp[3]		= system_clk_in_cnt[22];

always @ (posedge system_clk_in_g or negedge reset_in_n)
begin
	if (~reset_in_n)
		system_clk_in_cnt	<= 26'h0;
	else
		system_clk_in_cnt	<= system_clk_in_cnt + 26'h1;
end

always @ (posedge system_clk or negedge reset_n)
begin
	if (~reset_n)
		system_clk_cnt		<= 26'h0;
	else
		system_clk_cnt		<= system_clk_cnt + 26'h1;
end

sys_dcm			sys_dcm_inst
(
		.CLKIN_IN				(system_clk_in),
	  .RST_IN					(reset),
	  .CLKDV_OUT				(system_clk),
	  .CLKIN_IBUFG_OUT		(system_clk_in_g),
	  .CLK0_OUT				(),
		.CLK2X_OUT				(system_clk_2x),
	  .LOCKED_OUT				(sys_clk_locked)
);

`endif

///			DVI

`ifdef		__DVI_SIM__

wire	[7:0]	dvi_r_sim;
wire	[7:0]	dvi_g_sim;
wire	[7:0]	dvi_b_sim;
wire			dvi_dclk_sim;
wire			dvi_de_sim;
wire			dvi_vsync_sim;
wire			dvi_hsync_sim;

dvi_sim		dvi_sim_inst
(
	.r				(dvi_r_sim),
	.g				(dvi_g_sim),
	.b				(dvi_b_sim),
	.dclk		(dvi_dclk_sim),
	.de				(dvi_de_sim),
	.vsync		(dvi_vsync_sim),
	.hsync		(dvi_hsync_sim),

	.clk		(system_clk),

	.reset_n(reset_n)
);

`endif

dvi_to_ram		dvi_to_ram_inst
(
`ifdef		__DVI_SIM__
	.r					(dvi_r_sim),
	.g					(dvi_g_sim),
	.b					(dvi_b_sim),
	.dclk				(dvi_dclk_sim),
	.de					(dvi_de_sim),
	.vsync				(dvi_vsync_sim),
	.hsync				(dvi_hsync_sim),
`else
	.r					(dvi_r),
	.g					(dvi_g),
		.b					(dvi_b),
		.dclk				(dvi_dclk),
		.de					(dvi_de),
		.vsync				(dvi_vsync),
		.hsync				(dvi_hsync),
`endif

		.ram_b0b6_wr_en		(ram_b0b6_wr_en),
		.ram_b7_wr_en	(ram_b7_wr_en),
		.ram_wr_data	(ram_wr_data),
		.ram_wr_addr	(ram_wr_addr),

		.disp_req			(disp_req),
		.disp_ack			(disp_ack),
		.disp_buff_sel	(disp_buff_sel),

		.reset_n		(reset_n)
);

///			output module

//assign		a[1]	= a[0];
//assign		b[1]	= b[0];
//assign		c[1]	= c[0];
//assign		d[1]	= d[0];

//assign		clk[1]	= clk[0];
//assign		latch[1]= latch[0];
//assign		oe_n[1]= oe_n[0];

//assign		b_serial[0]		= 0;
//assign		b_serial[1]		= 0;
//assign		b_serial_s[0]	= 0;
//assign		b_serial_s[1]	= 0;

led_sub_module			sub_module_0
(
			.r_0					(r_serial[0]),
			.g_0					(g_serial[0]),
			.b_0					(b_serial[0]),

			.r_1					(r_serial[1]),
			.g_1					(g_serial[1]),
			.b_1					(b_serial[1]),

			.clk					(clk[0]),
			.latch					(latch[0]),
			.oe_n					(oe_n[0]),

			.a						(a[0]),
			.b						(b[0]),
			.c						(c[0]),
			.d						(d[0]),

			.r_0_second				(r_serial_s[0]),
			.g_0_second				(g_serial_s[0]),
			.b_0_second				(b_serial_s[0]),

			.r_1_second				(r_serial_s[1]),
			.g_1_second				(g_serial_s[1]),
			.b_1_second				(b_serial_s[1]),

			.clk_second				(clk_s[0]),
			.latch_second		(latch_s[0]),
			.oe_n_second		(oe_n_s[0]),

			.a_second				(a_s[0]),
			.b_second				(b_s[0]),
			.c_second				(c_s[0]),
			.d_second				(d_s[0]),

			.ram_wr_addr		(ram_wr_addr),
			.ram_wr_data		(ram_wr_data),
			.ram_b0b6_wr_en_0	(ram_b0b6_wr_en[0]),
			.ram_b0b6_wr_en_1	(ram_b0b6_wr_en[1]),
			.ram_b7_wr_en_0			(ram_b7_wr_en[0]),
			.ram_b7_wr_en_1			(ram_b7_wr_en[1]),

			.disp_req				(disp_req),
			.disp_ack				(disp_ack),
			.disp_buff_sel		(disp_buff_sel),

`ifdef		__DVI_SIM__
			.dclk					(dvi_dclk_sim),
`else
			.dclk					(dvi_dclk),
`endif

			.cs_data			(cs_data_module_0),

			.system_clk			(system_clk),
			.reset_n		(reset_n)
);

led_sub_module			sub_module_1
(
			.r_0				(r_serial[2]),
			.g_0				(g_serial[2]),
			.b_0				(b_serial[2]),

			.r_1				(r_serial[3]),
			.g_1				(g_serial[3]),
			.b_1				(b_serial[3]),

			.clk				(clk[1]),
			.latch				(latch[1]),
			.oe_n				(oe_n[1]),

			.a					(a[1]),
			.b					(b[1]),
			.c					(c[1]),
			.d					(d[1]),

			.r_0_second			(r_serial_s[2]),
			.g_0_second			(g_serial_s[2]),
			.b_0_second			(b_serial_s[2]),

			.r_1_second			(r_serial_s[3]),
			.g_1_second			(g_serial_s[3]),
			.b_1_second			(b_serial_s[3]),

			.clk_second			(clk_s[1]),
			.latch_second		(latch_s[1]),
			.oe_n_second		(oe_n_s[1]),

			.a_second			(a_s[1]),
			.b_second			(b_s[1]),
			.c_second			(c_s[1]),
			.d_second			(d_s[1]),

			.ram_wr_addr		(ram_wr_addr),
			.ram_wr_data		(ram_wr_data),
			.ram_b0b6_wr_en_0	(ram_b0b6_wr_en[2]),
			.ram_b0b6_wr_en_1	(ram_b0b6_wr_en[3]),
			.ram_b7_wr_en_0			(ram_b7_wr_en[2]),
			.ram_b7_wr_en_1			(ram_b7_wr_en[3]),

			.disp_req			(disp_req),
			.disp_ack			(),
			.disp_buff_sel		(disp_buff_sel),

`ifdef		__DVI_SIM__
			.dclk				(dvi_dclk_sim),
`else
			.dclk				(dvi_dclk),
`endif

			.system_clk			(system_clk),
			.reset_n		(reset_n)
);

led_sub_module			sub_module_2
(
			.r_0				(r_serial[4]),
			.g_0				(g_serial[4]),
			.b_0				(b_serial[4]),

			.r_1				(r_serial[5]),
			.g_1				(g_serial[5]),
			.b_1				(b_serial[5]),

			.clk				(clk[2]),
			.latch				(latch[2]),
			.oe_n				(oe_n[2]),

			.a					(a[2]),
			.b					(b[2]),
			.c					(c[2]),
			.d					(d[2]),

			.r_0_second			(r_serial_s[4]),
			.g_0_second			(g_serial_s[4]),
                .b_0_second                     (b_serial_s[4]),

                .r_1_second                     (r_serial_s[5]),
                .g_1_second                     (g_serial_s[5]),
                .b_1_second                     (b_serial_s[5]),

                .clk_second                     (clk_s[2]),
                .latch_second           (latch_s[2]),
                .oe_n_second            (oe_n_s[2]),

                .a_second                       (a_s[2]),
                .b_second                       (b_s[2]),
                .c_second                       (c_s[2]),
                .d_second                       (d_s[2]),

                .ram_wr_addr            (ram_wr_addr),
                .ram_wr_data            (ram_wr_data),
                .ram_b0b6_wr_en_0       (ram_b0b6_wr_en[4]),
                .ram_b0b6_wr_en_1       (ram_b0b6_wr_en[5]),
                .ram_b7_wr_en_0                 (ram_b7_wr_en[4]),
                .ram_b7_wr_en_1                 (ram_b7_wr_en[5]),

                .disp_req                       (disp_req),
                .disp_ack                       (),
                .disp_buff_sel          (disp_buff_sel),

`ifdef          __DVI_SIM__
                .dclk                           (dvi_dclk_sim),
`else
                .dclk                           (dvi_dclk),
`endif

                .system_clk                     (system_clk),
                .reset_n                (reset_n)
);

led_sub_module                  sub_module_3
(
                .r_0                            (r_serial[6]),
                .g_0                            (g_serial[6]),
                .b_0                            (b_serial[6]),

                .r_1                            (r_serial[7]),
                .g_1                            (g_serial[7]),
                .b_1                            (b_serial[7]),

                .clk                            (clk[3]),
                .latch                          (latch[3]),
                .oe_n                           (oe_n[3]),

                .a                                      (a[3]),
                .b                                      (b[3]),
                .c                                      (c[3]),
                .d                                      (d[3]),

                .r_0_second                     (r_serial_s[6]),
                .g_0_second                     (g_serial_s[6]),
                .b_0_second                     (b_serial_s[6]),

                .r_1_second                     (r_serial_s[7]),
                .g_1_second                     (g_serial_s[7]),
                .b_1_second                     (b_serial_s[7]),

                .clk_second                     (clk_s[3]),
                .latch_second           (latch_s[3]),
                .oe_n_second            (oe_n_s[3]),

                .a_second                       (a_s[3]),
                .b_second                       (b_s[3]),
                .c_second                       (c_s[3]),
                .d_second                       (d_s[3]),

                .ram_wr_addr            (ram_wr_addr),
                .ram_wr_data            (ram_wr_data),
                .ram_b0b6_wr_en_0       (ram_b0b6_wr_en[6]),
                .ram_b0b6_wr_en_1       (ram_b0b6_wr_en[7]),
                .ram_b7_wr_en_0                 (ram_b7_wr_en[6]),
                .ram_b7_wr_en_1                 (ram_b7_wr_en[7]),

                .disp_req                       (disp_req),
                .disp_ack                       (),
                .disp_buff_sel          (disp_buff_sel),

`ifdef          __DVI_SIM__
                .dclk                           (dvi_dclk_sim),
`else
```

```
		.dclk				(dvi_dclk),
`endif

		.system_clk			(system_clk),
		.reset_n			(reset_n)
);

led_sub_module			sub_module_4
(
		.r_0				(r_serial[8]),
		.g_0				(g_serial[8]),
		.b_0				(b_serial[8]),

		.r_1				(r_serial[9]),
		.g_1				(g_serial[9]),
		.b_1				(b_serial[9]),

		.clk				(clk[4]),
		.latch				(latch[4]),
		.oe_n				(oe_n[4]),

		.a					(a[4]),
		.b					(b[4]),
		.c					(c[4]),
		.d					(d[4]),

		.r_0_second			(r_serial_s[8]),
		.g_0_second			(g_serial_s[8]),
		.b_0_second			(b_serial_s[8]),

		.r_1_second			(r_serial_s[9]),
		.g_1_second			(g_serial_s[9]),
		.b_1_second			(b_serial_s[9]),

		.clk_second			(clk_s[4]),
		.latch_second		(latch_s[4]),
		.oe_n_second		(oe_n_s[4]),

		.a_second			(a_s[4]),
		.b_second			(b_s[4]),
		.c_second			(c_s[4]),
		.d_second			(d_s[4]),

		.ram_wr_addr		(ram_wr_addr),
		.ram_wr_data		(ram_wr_data),
		.ram_b0b6_wr_en_0	(ram_b0b6_wr_en[8]),
		.ram_b0b6_wr_en_1	(ram_b0b6_wr_en[9]),
		.ram_b7_wr_en_0			(ram_b7_wr_en[8]),
		.ram_b7_wr_en_1			(ram_b7_wr_en[9]),

		.disp_req			(disp_req),
		.disp_ack			(),
		.disp_buff_sel		(disp_buff_sel),

`ifdef	__DVI_SIM__
		.dclk				(dvi_dclk_sim),
`else
		.dclk				(dvi_dclk),
`endif

		.system_clk			(system_clk),
		.reset_n			(reset_n)
);

led_sub_module			sub_module_5
(
		.r_0				(r_serial[10]),
		.g_0				(g_serial[10]),
		.b_0				(b_serial[10]),

		.r_1				(r_serial[11]),
		.g_1				(g_serial[11]),
		.b_1				(b_serial[11]),

		.clk				(clk[5]),
		.latch				(latch[5]),
		.oe_n				(oe_n[5]),

		.a					(a[5]),
		.b					(b[5]),
		.c					(c[5]),
		.d					(d[5]),

		.r_0_second			(r_serial_s[10]),
		.g_0_second			(g_serial_s[10]),
		.b_0_second			(b_serial_s[10]),
		.r_1_second			(r_serial_s[11]),
		.g_1_second			(g_serial_s[11]),
		.b_1_second			(b_serial_s[11]),

		.clk_second			(clk_s[5]),
		.latch_second		(latch_s[5]),
		.oe_n_second		(oe_n_s[5]),

		.a_second			(a_s[5]),
		.b_second			(b_s[5]),
		.c_second			(c_s[5]),
		.d_second			(d_s[5]),

		.ram_wr_addr		(ram_wr_addr),
		.ram_wr_data		(ram_wr_data),
		.ram_b0b6_wr_en_0	(ram_b0b6_wr_en[10]),
		.ram_b0b6_wr_en_1	(ram_b0b6_wr_en[11]),
		.ram_b7_wr_en_0			(ram_b7_wr_en[10]),
		.ram_b7_wr_en_1			(ram_b7_wr_en[11]),

		.disp_req			(disp_req),
		.disp_ack			(),
		.disp_buff_sel		(disp_buff_sel),

`ifdef	__DVI_SIM__
		.dclk				(dvi_dclk_sim),
`else
		.dclk				(dvi_dclk),
`endif

		.system_clk			(system_clk),
		.reset_n		(reset_n)
);


///		logic analysis module

/*
wire	[35:0]	chipscope_controlline;
wire	[63:0]	chipscope_data;

//assign		chipscope_data = {ram_b0b6_wr_en[0], cs_data_module_0[254:0]};

assign		chipscope_data =
{
		dvi_b,dvi_g,dvi_r,
		dvi_hsync, dvi_vsync, dvi_de, dvi_dclk
};
*/
/*
assign		chipscope_data =
{
//		dvi_hsync, dvi_vsync, dvi_de, dvi_dclk
		ram_wr_data,ram_wr_addr,
		disp_buff_sel,disp_ack,disp_req,
		ram_b7_wr_en[1],ram_b7_wr_en[0],ram_b0b6_wr_en[1],ram_b0b6_wr_en[0],
		dvi_hsync_sim,dvi_vsync_sim,dvi_de_sim,dvi_dclk_sim
//		dvi_b_sim,dvi_g_sim,dvi_r_sim,
//		d[0],c[0],b[0],a[0],oe_n[0],latch[0],clk[0],b_serial[0],g_serial[0],r_serial[0]
};
*/
/*
chipscope_ila ila_inst
(
		.CLK			(system_clk_2x),
		.CONTROL		(chipscope_controlline),
		.TRIG0			(chipscope_data)
);

chipscope_icon icon_inst
(
		.CONTROL0		(chipscope_controlline)
);
*/
/*
wire	[35:0]	chipscope_controlline;
wire	[63:0]	chipscope_data;

assign		chipscope_data =
{
		latch_s, clk_r0_c0_s,
		ram_b0b6_rd_data_1[31:0],
		r_r0_c0_1_s, g_r0_c0_1_s, b_r0_c0_1_s,
		ram_b0b6_bitsel_status, ram_b0b6_rd_addr,
		r_r0_c0_0_s, g_r0_c0_0_s, b_r0_c0_0_s,
                a, b, c, d,
                latch, oe_n, clk_r0_c0,
                r_r0_c0_1, g_r0_c0_1, b_r0_c0_1, r_r0_c0_0, g_r0_c0_0, b_r0_c0_0
};

chipscope_ila ila_inst
(
                .CLK                    (system_clk),
                .CONTROL                (chipscope_controlline),
                .TRIG0                  (chipscope_data)
);

chipscope_icon icon_inst
(
                .CONTROL0               (chipscope_controlline)
);
*/

endmodule
```

2) Bit-plane separating module

```
`include                "../source/define.v"

/*                      definition
                                                                        */

module led_out_ptos
(
        sdi_r,
        sdi_g,
        sdi_b,

        sdi_clk,

        data_r,
        data_g,
        data_b,

        data_en,

        system_clk,
        reset_n
);

/*                      definition
                                                                        */

output                  sdi_r;
output                  sdi_g;
output                  sdi_b;

output                  sdi_clk;

input   [31:0]  data_r;
input   [31:0]  data_g;
input   [31:0]  data_b;

input                   data_en;

input                   system_clk;
input                   reset_n;

/*                      definition
                                                                        */

parameter       S_IDLE          = 2'h0,
                        S_OUT                   = 2'h1;

/*                      definition
                                                                        */

reg     [1:0]    state_next;
reg     [1:0]    state_cur;

reg             [31:0]  data_r_reg;
reg             [31:0]  data_g_reg;
reg             [31:0]  data_b_reg;

reg             [4:0]   bit_cnt;

wire                    bit_out_end;
wire                    st_idle;
wire                    st_out;

reg                             st_out_d1;

reg                             st_out_d2_neg;

wire                    system_clk_inv;
wire                    sdi_clk_en;

reg                             sdi_r;
reg                             sdi_g;
reg                             sdi_b;

/*                      main
                                                                        */

///             状态机

assign          st_idle                 = state_cur==S_IDLE;
assign          st_out                          = state_cur==S_OUT;

always@ (posedge system_clk or negedge reset_n)
begin
        if (~reset_n)
                state_cur                               <= S_IDLE;
        else
                state_cur                       <= state_next;
end

always@ (*)
begin

        state_next                              <= state_cur;

        case (state_cur)
                S_IDLE:
                        if (data_en)
                                state_next      <= S_OUT;

                S_OUT:
                        if (bit_out_end & (~data_en))
                                state_next      <= S_IDLE;

                default:
                        state_next              <= S_IDLE;
        endcase
end

///             data input

always@ (posedge system_clk or negedge reset_n)
begin
        if (~reset_n)
                begin
                        data_r_reg                      <= 32'h0;
                        data_g_reg                      <= 32'h0;
                        data_b_reg                      <= 32'h0;
                end
        else if ((data_en & bit_out_end) | st_idle)
                begin
                        data_r_reg                      <= {data_r[15:0], data_r[31:16]};
                        data_g_reg                      <= {data_g[15:0], data_g[31:16]};
                        data_b_reg                      <= {data_b[15:0], data_b[31:16]};
                end
end

///             data output

assign          system_clk_inv          = (~system_clk);
assign          sdi_clk_en                      = st_out_d2_neg & st_out_d1;
assign          sdi_clk                         = sdi_clk_en & system_clk_inv;

assign          bit_out_end                     = bit_cnt==5'h1f;

always@ (posedge system_clk or negedge reset_n)
begin
        if (~reset_n)
                st_out_d1                               <= 1'b0;
        else
                st_out_d1                       <= st_out;
end

always@ (negedge system_clk or negedge reset_n)
begin
        if (~reset_n)
```

```
			st_out_d2_neg				<= 1'b0;
		else
			st_out_d2_neg					<= st_out_d1;
end

always@ (posedge system_clk or negedge reset_n)
begin
		if (~reset_n)
			bit_cnt					<= 5'h0;
		else if (st_idle)
			bit_cnt					<= 5'h0;
		else
			bit_cnt					<= bit_cnt + 5'h1;
end

always@ (posedge system_clk or negedge reset_n)
begin
		if (~reset_n)
			begin
				sdi_r				<= 1'b0;
				sdi_g				<= 1'b0;
				sdi_b				<= 1'b0;
			end
		else if (st_idle)
			begin
				sdi_r				<= 1'b0;
				sdi_g				<= 1'b0;
				sdi_b				<= 1'b0;
			end
		else
			case (bit_cnt)
				5'd0	: begin
							sdi_r	<= data_r_reg[0];
							sdi_g	<= data_g_reg[0];
							sdi_b	<= data_b_reg[0];
						end
				5'd1	: begin
							sdi_r	<= data_r_reg[1];
							sdi_g	<= data_g_reg[1];
							sdi_b	<= data_b_reg[1];
						end
				5'd2	: begin
							sdi_r	<= data_r_reg[2];
							sdi_g	<= data_g_reg[2];
							sdi_b	<= data_b_reg[2];
						end
				5'd3	: begin
							sdi_r	<= data_r_reg[3];
							sdi_g	<= data_g_reg[3];
							sdi_b	<= data_b_reg[3];
						end
				5'd4	: begin
							sdi_r	<= data_r_reg[4];
							sdi_g	<= data_g_reg[4];
							sdi_b	<= data_b_reg[4];
						end
				5'd5	: begin
							sdi_r	<= data_r_reg[5];
							sdi_g	<= data_g_reg[5];
							sdi_b	<= data_b_reg[5];
						end
				5'd6	: begin
							sdi_r	<= data_r_reg[6];
							sdi_g	<= data_g_reg[6];
							sdi_b	<= data_b_reg[6];
						end
				5'd7	: begin
							sdi_r	<= data_r_reg[7];
							sdi_g	<= data_g_reg[7];
							sdi_b	<= data_b_reg[7];
						end
				5'd8	: begin
							sdi_r	<= data_r_reg[8];
							sdi_g	<= data_g_reg[8];
							sdi_b	<= data_b_reg[8];
						end
				5'd9	: begin
							sdi_r	<= data_r_reg[9];
							sdi_g	<= data_g_reg[9];
							sdi_b	<= data_b_reg[9];
						end
				5'd10 :  begin
							sdi_r	<= data_r_reg[10];
							sdi_g	<= data_g_reg[10];
							sdi_b	<= data_b_reg[10];
						end
```

```
5'd11 :  begin
                    sdi_r     <= data_r_reg[11];
                    sdi_g     <= data_g_reg[11];
                    sdi_b     <= data_b_reg[11];
             end
5'd12 :  begin
                    sdi_r     <= data_r_reg[12];
                    sdi_g     <= data_g_reg[12];
                    sdi_b     <= data_b_reg[12];
             end
5'd13 :  begin
                    sdi_r     <= data_r_reg[13];
                    sdi_g     <= data_g_reg[13];
                    sdi_b     <= data_b_reg[13];
             end
5'd14    :       begin
                    sdi_r     <= data_r_reg[14];
                    sdi_g     <= data_g_reg[14];
                    sdi_b     <= data_b_reg[14];
             end
5'd15    :       begin
                    sdi_r     <= data_r_reg[15];
                    sdi_g     <= data_g_reg[15];
                    sdi_b     <= data_b_reg[15];
             end
5'd16 :  begin
                    sdi_r     <= data_r_reg[16];
                    sdi_g     <= data_g_reg[16];
                    sdi_b     <= data_b_reg[16];
             end
5'd17 :  begin
                    sdi_r     <= data_r_reg[17];
                    sdi_g     <= data_g_reg[17];
                    sdi_b     <= data_b_reg[17];
             end
5'd18 :  begin
                    sdi_r     <= data_r_reg[18];
                    sdi_g     <= data_g_reg[18];
                    sdi_b     <= data_b_reg[18];
             end
5'd19 :  begin
                    sdi_r     <= data_r_reg[19];
                    sdi_g     <= data_g_reg[19];
                    sdi_b     <= data_b_reg[19];
             end
5'd20 :  begin
                    sdi_r     <= data_r_reg[20];
                    sdi_g     <= data_g_reg[20];
                    sdi_b     <= data_b_reg[20];
             end
5'd21 :  begin
                    sdi_r     <= data_r_reg[21];
                    sdi_g     <= data_g_reg[21];
                    sdi_b     <= data_b_reg[21];
             end
5'd22 :  begin
                    sdi_r     <= data_r_reg[22];
                    sdi_g     <= data_g_reg[22];
                    sdi_b     <= data_b_reg[22];
             end
5'd23 :  begin
                    sdi_r     <= data_r_reg[23];
                    sdi_g     <= data_g_reg[23];
                    sdi_b     <= data_b_reg[23];
             end
5'd24 :  begin
                    sdi_r     <= data_r_reg[24];
                    sdi_g     <= data_g_reg[24];
                    sdi_b     <= data_b_reg[24];
             end
5'd25 :  begin
                    sdi_r     <= data_r_reg[25];
                    sdi_g     <= data_g_reg[25];
                    sdi_b     <= data_b_reg[25];
             end
5'd26 :  begin
                    sdi_r     <= data_r_reg[26];
                    sdi_g     <= data_g_reg[26];
                    sdi_b     <= data_b_reg[26];
             end
5'd27 :  begin
                    sdi_r     <= data_r_reg[27];
                    sdi_g     <= data_g_reg[27];
                    sdi_b     <= data_b_reg[27];
             end
5'd28 :  begin
```

```
                                                sdi_r    <= data_r_reg[28];
                                                sdi_g    <= data_g_reg[28];
                                                sdi_b    <= data_b_reg[28];
                                        end
                        5'd29 :  begin
                                                sdi_r    <= data_r_reg[29];
                                                sdi_g    <= data_g_reg[29];
                                                sdi_b    <= data_b_reg[29];
                                        end
                        5'd30 :  begin
                                                sdi_r    <= data_r_reg[30];
                                                sdi_g    <= data_g_reg[30];
                                                sdi_b    <= data_b_reg[30];
                                        end
                        5'd31 :  begin
                                                sdi_r    <= data_r_reg[31];
                                                sdi_g    <= data_g_reg[31];
                                                sdi_b    <= data_b_reg[31];
                                        end
                endcase
end

endmodule
```

3) Scan controller module

```
/*                      definition
                                                                  */

`include                "../source/define.v"

module led_out_ctrl
(
        a,
     b,
     c,
     d,

     latch,
     oe_n,

        ram_rd_addr,

        ram_b0b6_rd_data_0,
        ram_b0b6_rd_data_1,
        ram_b7_rd_data_0,
        ram_b7_rd_data_1,

        disp_req,
        disp_ack,
        disp_buff_sel,

        data_r_0,
        data_g_0,
        data_b_0,

        data_r_1,
        data_g_1,
        data_b_1,

        data_en,

        data_r_second_0,
        data_g_second_0,
        data_b_second_0,

        data_r_second_1,
        data_g_second_1,
        data_b_second_1,

        data_en_second,

        system_clk,
        reset_n
);

/*                      definition
                                                                  */

output                  a;
output                  b;
output                  c;
output                  d;
output              latch;
output              oe_n;

output  [10:0]  ram_rd_addr;

input   [95:0]  ram_b0b6_rd_data_0;
input   [95:0]  ram_b0b6_rd_data_1;
input   [95:0]  ram_b7_rd_data_0;
input   [95:0]  ram_b7_rd_data_1;

input               disp_req;
output              disp_ack;
input               disp_buff_sel;

output  [31:0]  data_r_0;
output  [31:0]  data_g_0;
output  [31:0]  data_b_0;
output  [31:0]  data_r_1;
output  [31:0]  data_g_1;
output  [31:0]  data_b_1;
output              data_en;

output  [31:0]  data_r_second_0;
output  [31:0]  data_g_second_0;
output  [31:0]  data_b_second_0;
output  [31:0]  data_r_second_1;
output  [31:0]  data_g_second_1;
output  [31:0]  data_b_second_1;
output              data_en_second;

input               system_clk;
input               reset_n;

/*                  definition
                                                */

parameter       S_IDLE                      = 3'h0,
                    S_BIT0_ROW0_DATA_OUT    = 3'h1,
                    S_BIT0_ROW0_WAIT        = 3'h2,
                    S_ROW_DATA_LATCH            = 3'h3,
                    S_NEXT_ROW_DATA_OUT         = 3'h4,
                    S_FRAME_TIME_WAIT           = 3'h5,
                    S_ACK                       = 3'h6,
                    S_ROW_TIME_WAIT             = 3'h7;

`ifdef          __OUT_SLOW__

parameter       TICK_CNT_MAX                = 24'hffff;
`ifdef   __OE_TIME_CTRL__
parameter       OE_TIME_CNT_MIN             = 24'h10;
parameter       OE_TIME_CNT_MAX             = 24'h11;
`endif

`else

//parameter     TICK_CNT_MAX                = 9'd320;
parameter       TICK_CNT_MAX                = 9'd270;
`ifdef   __OE_TIME_CTRL__
parameter       OE_TIME_CNT_MIN             = 9'd10;
parameter       OE_TIME_CNT_MAX             = 9'd11;
`endif

`endif

//parameter     FRAME_TICK_CNT_MAX          = 20'd332232;
parameter       FRAME_TICK_CNT_MAX          = 20'd664464;
//此参数为每行的模块数目-1
parameter       SUB_MODULE_NUM              = 3'd7;

/*                  definition
                                                */

reg             [3:0]   row_cnt;                        //
reg             [4:0]   serial_out_bit_cnt;         //
reg             [2:0]   serial_out_word_cnt;    //
reg             [3:0]   serial_out_row_cnt;         //
reg             [2:0]   serial_out_b0b6_cnt;    //

wire                    is_one_word_out_end;  //
wire                    is_one_row_out_end;         //
wire                    is_one_scr_out_end;         //
wire                    is_data_out_going;          //


reg             [2:0]   b0b6_cnt;                       //
wire                    is_b0b6_1bit_disp_end;//
```

```
`ifdef		__OUT_SLOW__
reg		[23:0]	tick_cnt;				//
`else
reg		[8:0]	tick_cnt;				//
`endif

wire			is_frame_time_end;
reg		[19:0]	frame_tick_cnt;

reg		[5:0]	piece_cnt;				//
reg		[5:0]	row_time_piece_max;		//

wire			is_one_piece_end;		//
wire			is_piece_cnt_max;		//
wire			is_row_time_end;		//
wire			is_one_frame_end;		//

reg				disp_buff_sel_reg;
reg				disp_req_reg;

reg		[2:0]	bit0_row0_wait_cnt;
wire			is_bit0_row0_wait_end;

reg				is_data_out_going_d1;
reg				st_row_data_latch_d1;

wire			st_idle;			//	state machine
wire			st_bit0_row0_data_out;
wire			st_bit0_row0_wait;
wire			st_row_data_latch;
wire			st_next_row_data_out;
wire			st_frame_time_wait;
wire			st_ack;

reg	[2:0]	state_next;
reg	[2:0]	state_cur;

/*			main
							*/

assign		disp_ack			= st_ack;

///

assign		a				= row_cnt[0];
assign		b				= row_cnt[1];
assign		c				= row_cnt[2];
assign		d				= ~row_cnt[3];

`ifdef	__OE_TIME_CTRL__

`ifdef	__FRAME_TIME_CTRL__
assign		oe_n			= st_idle | st_bit0_row0_data_out | st_bit0_row0_wait | st_frame_time_wait |
						(tick_cnt<OE_TIME_CNT_MIN) | (tick_cnt>OE_TIME_CNT_MAX);
`else
assign		oe_n			= st_idle | st_bit0_row0_data_out | st_bit0_row0_wait |
						(tick_cnt<OE_TIME_CNT_MIN) | (tick_cnt>OE_TIME_CNT_MAX);
`endif

`else

`ifdef	__FRAME_TIME_CTRL__
assign		oe_n			= st_idle | st_bit0_row0_data_out | st_bit0_row0_wait | st_frame_time_wait;
`else
assign		oe_n			= st_idle | st_bit0_row0_data_out | st_bit0_row0_wait;
`endif

`endif

assign		latch			= st_row_data_latch | st_row_data_latch_d1;

///

assign		data_en			= is_data_out_going_d1;
assign		data_r_0		= ram_b0b6_rd_data_0[31:0];
assign		data_g_0		= ram_b0b6_rd_data_0[63:32];
assign		data_b_0		= ram_b0b6_rd_data_0[95:64];
assign		data_r_1		= ram_b0b6_rd_data_1[31:0];
assign		data_g_1		= ram_b0b6_rd_data_1[63:32];
assign		data_b_1		= ram_b0b6_rd_data_1[95:64];

assign		data_en_second		= is_data_out_going_d1;
assign		data_r_second_0		= ram_b7_rd_data_0[31:0];
assign		data_g_second_0		= ram_b7_rd_data_0[63:32];
assign		data_b_second_0		= ram_b7_rd_data_0[95:64];
```

```
assign          data_r_second_1                 = ram_b7_rd_data_1[31:0];
assign          data_g_second_1                 = ram_b7_rd_data_1[63:32];
assign          data_b_second_1                 = ram_b7_rd_data_1[95:64];

always@ (posedge system_clk or negedge reset_n)
begin
        if (~reset_n)
                begin
                        is_data_out_going_d1    <= 1'b0;
                        st_row_data_latch_d1    <= 1'b0;

                        disp_buff_sel_reg               <= 1'b0;
                        disp_req_reg                    <= 1'b0;
                end
        else
                begin
                        is_data_out_going_d1    <= is_data_out_going;
                        st_row_data_latch_d1    <= st_row_data_latch;

                        disp_buff_sel_reg               <= disp_buff_sel;
                        disp_req_reg                    <= disp_req;
                end
end

///             RAM 接口输出

assign          ram_rd_addr                     = {disp_buff_sel_reg, serial_out_b0b6_cnt, serial_out_row_cnt, serial_out_word_cnt};

///             RAM 接口处理

//
assign          is_data_out_going       = st_bit0_row0_data_out | st_next_row_data_out;
//
assign          is_one_word_out_end     = serial_out_bit_cnt==5'h1f;
//
assign          is_one_row_out_end      = (serial_out_word_cnt==SUB_MODULE_NUM) & is_one_word_out_end;
//
assign          is_one_scr_out_end      = (serial_out_row_cnt==4'hf) & is_one_row_out_end;

//
always@ (posedge system_clk or negedge reset_n)
begin
        if (~reset_n)
                serial_out_bit_cnt              <= 5'h0;
        else if (is_data_out_going)
                serial_out_bit_cnt              <= serial_out_bit_cnt + 5'h1;
        else
                serial_out_bit_cnt              <= 5'h0;
end

//
always@ (posedge system_clk or negedge reset_n)
begin
        if (~reset_n)
                serial_out_word_cnt             <= 3'h0;
        else if (is_data_out_going)
                begin
                        if (is_one_word_out_end)
                                serial_out_word_cnt     <= serial_out_word_cnt + 3'h1;
                end
        else
                serial_out_word_cnt             <= 3'h0;
end

//
always@ (posedge system_clk or negedge reset_n)
begin
        if (~reset_n)
                serial_out_row_cnt              <= 4'h0;
        else if (st_idle)
                serial_out_row_cnt              <= 4'h0;
        else if (is_one_row_out_end)
                serial_out_row_cnt              <= serial_out_row_cnt + 4'h1;
end

//
//
always@ (posedge system_clk or negedge reset_n)
begin
        if (~reset_n)
                serial_out_b0b6_cnt                     <= 3'h0;
        else if (st_idle)
                serial_out_b0b6_cnt                     <= 3'h0;
        else if (is_one_scr_out_end)
                begin
                        if (serial_out_b0b6_cnt==3'h6)
```

```
                              serial_out_b0b6_cnt      <= 3'h0;
                        else
                              serial_out_b0b6_cnt      <= serial_out_b0b6_cnt + 3'h1;
                  end
end

///

//
assign            is_b0b6_1bit_disp_end = (row_cnt==4'hf) & st_row_data_latch;

//
//
always@ (posedge system_clk or negedge reset_n)
begin
      if (~reset_n)
            row_cnt                                   <= 4'hf;
      else if (st_idle)
            row_cnt                                   <= 4'hf;
      else if (st_row_data_latch)
            row_cnt                                   <= row_cnt + 4'h1;
end

//
//
always@ (posedge system_clk or negedge reset_n)
begin
      if (~reset_n)
            b0b6_cnt                           <= 3'h6;
      else if (st_idle)
            b0b6_cnt                           <= 3'h6;
      else if (is_b0b6_1bit_disp_end)
            begin
                  if (b0b6_cnt==3'h6)
                        b0b6_cnt               <= 3'h0;
                  else
                        b0b6_cnt               <= b0b6_cnt + 3'h1;
            end
end

///

//
assign            is_one_piece_end        = tick_cnt==TICK_CNT_MAX;
//
assign            is_piece_cnt_max        = piece_cnt==row_time_piece_max;
//
assign            is_row_time_end               = is_piece_cnt_max & is_one_piece_end;
//
assign            is_one_frame_end        = (b0b6_cnt==3'h6) & (row_cnt==4'hf) & is_row_time_end;

//
always@ (*)
begin
      case (b0b6_cnt)
            3'h1     :      row_time_piece_max   <= (6'd2   - 6'd1);
            3'h2     :      row_time_piece_max   <= (6'd4   - 6'd1);
            3'h3     :      row_time_piece_max   <= (6'd8   - 6'd1);
            3'h4     :      row_time_piece_max   <= (6'd16 - 6'd1);
            3'h5     :      row_time_piece_max   <= (6'd32 - 6'd1);
            3'h6     :      row_time_piece_max   <= (6'd63);
            default  :      row_time_piece_max   <= (6'd1   - 6'd1);
      endcase
end

`ifdef            __OUT_SLOW__

//
always@ (posedge system_clk or negedge reset_n)
begin
      if (~reset_n)
            tick_cnt                    <= 24'h0;
      else if (st_idle | st_bit0_row0_data_out | st_bit0_row0_wait)
            tick_cnt                    <= 24'h0;
      else if (is_one_piece_end)
            tick_cnt                    <= 24'h0;
      else
            tick_cnt                    <= tick_cnt + 24'h1;
end

`else

//
always@ (posedge system_clk or negedge reset_n)
begin
      if (~reset_n)
```

```
			tick_cnt				<= 9'h0;
		else if (st_idle | st_bit0_row0_data_out | st_bit0_row0_wait)
			tick_cnt				<= 9'h0;
		else if (is_one_piece_end)
			tick_cnt				<= 9'h0;
		else
			tick_cnt				<= tick_cnt + 9'h1;
end

`endif

//
always@ (posedge system_clk or negedge reset_n)
begin
		if (~reset_n)
			piece_cnt					<= 6'h0;
		else if (st_idle | st_bit0_row0_data_out | st_bit0_row0_wait)
			piece_cnt					<= 6'h0;
		else if (is_one_piece_end)
			begin
				if (is_piece_cnt_max)
					piece_cnt			<= 6'h0;
				else
					piece_cnt			<= piece_cnt + 6'h1;
			end
end

///

assign		is_frame_time_end	= frame_tick_cnt==FRAME_TICK_CNT_MAX;

//
always@ (posedge system_clk or negedge reset_n)
begin
		if (~reset_n)
			frame_tick_cnt			<= 20'h0;
		else if (st_idle | st_bit0_row0_data_out | st_bit0_row0_wait)
			frame_tick_cnt			<= 20'h0;
		else if (is_frame_time_end)
			frame_tick_cnt			<= 20'h0;
		else
			frame_tick_cnt			<= frame_tick_cnt + 20'h1;
end

///

assign				is_bit0_row0_wait_end = bit0_row0_wait_cnt==3'h7;

always@ (posedge system_clk or negedge reset_n)
begin
		if (~reset_n)
			bit0_row0_wait_cnt		<= 3'h0;
		else if (st_bit0_row0_wait)
			bit0_row0_wait_cnt		<= bit0_row0_wait_cnt + 3'h1;
		else
			bit0_row0_wait_cnt		<= 3'h0;
end

///		state machine

assign		st_idle				= state_cur==S_IDLE;
assign		st_bit0_row0_data_out  = state_cur==S_BIT0_ROW0_DATA_OUT;
assign		st_bit0_row0_wait		= state_cur==S_BIT0_ROW0_WAIT;
assign		st_row_data_latch		= state_cur==S_ROW_DATA_LATCH;
assign		st_next_row_data_out   = state_cur==S_NEXT_ROW_DATA_OUT;
assign		st_frame_time_wait		= state_cur==S_FRAME_TIME_WAIT;
assign		st_ack				= state_cur==S_ACK;

always@ (posedge system_clk or negedge reset_n)
begin
		if (~reset_n)
			state_cur					<= S_IDLE;
		else
			state_cur				<= state_next;
end

`ifndef				__DATA_STATIC__

always@ (*)
begin

		state_next					= state_cur;

		case (state_cur)
			S_IDLE:
				if (disp_req_reg)
```

```
                        state_next      = S_ACK;

                S_ACK:
                    if (~disp_req_reg)
                        state_next      = S_BIT0_ROW0_DATA_OUT;

                S_BIT0_ROW0_DATA_OUT:
                    if (is_one_row_out_end)
                        state_next      = S_BIT0_ROW0_WAIT;

                S_BIT0_ROW0_WAIT:
                    if (is_bit0_row0_wait_end)
                        state_next      = S_ROW_DATA_LATCH;

                S_ROW_DATA_LATCH:
                    state_next          = S_NEXT_ROW_DATA_OUT;

                S_NEXT_ROW_DATA_OUT:
                    if (is_one_row_out_end)
                        state_next      = S_ROW_TIME_WAIT;

                S_ROW_TIME_WAIT:
                    if (is_row_time_end)
                        begin
                            if (is_one_frame_end)
                                state_next      = S_IDLE;
                            else
                                state_next      = S_ROW_DATA_LATCH;
                        end

                default:
                    state_next          = S_IDLE;
        endcase
end

`else

always@ (*)
begin

        state_next                      = state_cur;

        case (state_cur)
                S_IDLE:
                    state_next          = S_BIT0_ROW0_DATA_OUT;

                S_BIT0_ROW0_DATA_OUT:
                    if (is_one_row_out_end)
                        state_next      = S_BIT0_ROW0_WAIT;

                S_BIT0_ROW0_WAIT:
                    if (is_bit0_row0_wait_end)
                        state_next      = S_ROW_DATA_LATCH;

                S_ROW_DATA_LATCH:
                    state_next          = S_NEXT_ROW_DATA_OUT;

                S_NEXT_ROW_DATA_OUT:
                    if (is_one_row_out_end)
                        state_next      = S_ROW_TIME_WAIT;

                S_ROW_TIME_WAIT:
                    if (is_row_time_end)
                        begin
                            if (is_one_frame_end)
`ifdef  __FRAME_TIME_CTRL__
                                state_next      = S_FRAME_TIME_WAIT;
`else
                                state_next      = S_IDLE;
`endif
                            else
                                state_next      = S_ROW_DATA_LATCH;
                        end

                S_FRAME_TIME_WAIT:
                    if (is_frame_time_end)
                        state_next    = S_IDLE;

                default:
                    state_next          = S_IDLE;
        endcase
end

`endif

Endmodule
```

4) Driving modules

led_sub_module

```
`include                    "../source/define.v"

/*                          definition
                                                                      */

module led_sub_module
(
        r_0,
        g_0,
        b_0,

        r_1,
        g_1,
        b_1,

        clk,
        latch,
        oe_n,

        a,
        b,
        c,
        d,

        r_0_second,
        g_0_second,
        b_0_second,

        r_1_second,
        g_1_second,
        b_1_second,

        clk_second,
        latch_second,
        oe_n_second,

        a_second,
        b_second,
        c_second,
        d_second,

        ram_wr_addr,
        ram_wr_data,
        ram_b0b6_wr_en_0,
        ram_b0b6_wr_en_1,
        ram_b7_wr_en_0,
        ram_b7_wr_en_1,

        disp_req,
        disp_ack,
        disp_buff_sel,

        dclk,

        cs_data,

        system_clk,
        reset_n
);

/*                          definition
                                                                      */

output                      r_0;
output                      g_0;
output                      b_0;

output                      r_1;
output                      g_1;
output                      b_1;

output                      clk;
output                      latch;
output                      oe_n;

output                      a;
output                      b;
output                      c;
output                      d;
output                  r_0_second;
output                  g_0_second;
output                  b_0_second;

output                  r_1_second;
output                  g_1_second;
output                  b_1_second;

output                  clk_second;
output                  latch_second;
output                  oe_n_second;

output                  a_second;
output                  b_second;
output                  c_second;
output                  d_second;

input   [10:0]  ram_wr_addr;
input   [95:0]  ram_wr_data;
input                   ram_b0b6_wr_en_0;
input                   ram_b0b6_wr_en_1;
input                   ram_b7_wr_en_0;
input                   ram_b7_wr_en_1;

input                   disp_req;
output                  disp_ack;
input                   disp_buff_sel;

input                   dclk;

output  [255:0] cs_data;

input                   system_clk;
input                   reset_n;

/*                      definition
                                                                  */

wire    [31:0]  data_r_0;
wire    [31:0]  data_g_0;
wire    [31:0]  data_b_0;
wire    [31:0]  data_r_1;
wire    [31:0]  data_g_1;
wire    [31:0]  data_b_1;
wire                    data_en;

wire    [31:0]  data_r_second_0;
wire    [31:0]  data_g_second_0;
wire    [31:0]  data_b_second_0;
wire    [31:0]  data_r_second_1;
wire    [31:0]  data_g_second_1;
wire    [31:0]  data_b_second_1;
wire                    data_en_second;

wire    [10:0]  ram_rd_addr;

wire    [95:0]  ram_b0b6_rd_data_0;
wire    [95:0]  ram_b0b6_rd_data_1;
wire    [95:0]  ram_b7_rd_data_0;
wire    [95:0]  ram_b7_rd_data_1;

/*                      main
                                                           */

assign                  a_second                = a;
assign                  b_second                = b;
assign                  c_second                = c;
assign                  d_second                = d;

assign                  clk_second              = clk;
assign                  latch_second    = latch;
assign                  oe_n_second             = oe_n;

assign                  cs_data                 =
{
        ram_b0b6_rd_data_0,
        ram_rd_addr,
        ram_wr_data,
        ram_wr_addr,
        8'h0,
        ram_b0b6_wr_en_0,
        data_en
};

led_out_ctrl    led_out_ctrl_inst
```

```
(
		.a						(a),
	.b						(b),
	.c						(c),
	.d						(d),

	.latch						(latch),
	.oe_n						(oe_n),

		.ram_rd_addr				(ram_rd_addr),
		.ram_b0b6_rd_data_0			(ram_b0b6_rd_data_0),
		.ram_b0b6_rd_data_1			(ram_b0b6_rd_data_1),
		.ram_b7_rd_data_0			(ram_b7_rd_data_0),
		.ram_b7_rd_data_1			(ram_b7_rd_data_1),

		.disp_req					(disp_req),
		.disp_ack					(disp_ack),
		.disp_buff_sel				(disp_buff_sel),

		.data_r_0					(data_r_0),
		.data_g_0					(data_g_0),
		.data_b_0					(data_b_0),
		.data_r_1					(data_r_1),
		.data_g_1					(data_g_1),
		.data_b_1					(data_b_1),

		.data_en					(data_en),

		.data_r_second_0			(data_r_second_0),
		.data_g_second_0			(data_g_second_0),
		.data_b_second_0			(data_b_second_0),
		.data_r_second_1			(data_r_second_1),
		.data_g_second_1			(data_g_second_1),
		.data_b_second_1			(data_b_second_1),

		.data_en_second				(data_en_second),

		.system_clk					(system_clk),
		.reset_n				(reset_n)
);

led_out_ptos	led_out_ptos_0
(
		.sdi_r				(r_0),
		.sdi_g				(g_0),
		.sdi_b				(b_0),

		.sdi_clk			(clk),

		.data_r				(data_r_0),
		.data_g				(data_g_0),
		.data_b				(data_b_0),

		.data_en			(data_en),

		.system_clk			(system_clk),
		.reset_n			(reset_n)
);

led_out_ptos	led_out_ptos_1
(
		.sdi_r				(r_1),
		.sdi_g				(g_1),
		.sdi_b				(b_1),

		.sdi_clk			(),

		.data_r				(data_r_1),
		.data_g				(data_g_1),
		.data_b				(data_b_1),

		.data_en			(data_en),

		.system_clk			(system_clk),
		.reset_n			(reset_n)
);

`ifdef		__ONLY_ONE_CHANNEL__

assign		r_0_second	= r_0;
assign		g_0_second	= g_0;
assign		b_0_second	= b_0;

//assign		r_0_second	= 1'b1;
//assign		g_0_second	= 1'b1;
//assign		b_0_second	= 1'b1;
```

```
assign		r_1_second	= r_1;
assign		g_1_second	= g_1;
assign		b_1_second	= b_1;

`else

led_out_ptos	led_out_ptos_second_0
(
	.sdi_r				(r_0_second),
	.sdi_g				(g_0_second),
	.sdi_b				(b_0_second),

	.sdi_clk			(),

	.data_r				(data_r_second_0),
	.data_g				(data_g_second_0),
	.data_b				(data_b_second_0),

	.data_en			(data_en_second),

	.system_clk			(system_clk),
	.reset_n			(reset_n)
);

led_out_ptos	led_out_ptos_second_1
(
	.sdi_r				(r_1_second),
	.sdi_g				(g_1_second),
	.sdi_b				(b_1_second),

	.sdi_clk			(),

	.data_r				(data_r_second_1),
	.data_g				(data_g_second_1),
	.data_b				(data_b_second_1),

	.data_en			(data_en_second),

	.system_clk			(system_clk),
	.reset_n			(reset_n)
);

`endif

/*
		ALTERA RAM

*/

`ifdef	__ALTERA__

///		0 组 b0b6

b0b6_ram			b0b6_ram_r0
(
		.wrclock		(dclk),
		.wren			(ram_b0b6_wr_en_0),
		.wraddress		(ram_wr_addr),
		.data			(ram_wr_data[31:0]),

		.rdclock		(system_clk),
		.rdaddress		(ram_rd_addr),
		.q				(ram_b0b6_rd_data_0[31:0])
);

b0b6_ram			b0b6_ram_g0
(
		.wrclock		(dclk),
		.wren			(ram_b0b6_wr_en_0),
		.wraddress		(ram_wr_addr),
		.data			(ram_wr_data[63:32]),

		.rdclock		(system_clk),
		.rdaddress		(ram_rd_addr),
		.q				(ram_b0b6_rd_data_0[63:32])
);

b0b6_ram			b0b6_ram_b0
(
		.wrclock		(dclk),
		.wren			(ram_b0b6_wr_en_0),
		.wraddress		(ram_wr_addr),
		.data			(ram_wr_data[95:64]),

		.rdclock		(system_clk),
			.rdaddress			(ram_rd_addr),
			.q					(ram_b0b6_rd_data_0[95:64])
);

///			0 组 b7

b7_ram				b7_ram_r0
(
			.wrclock			(dclk),
			.wren				(ram_b7_wr_en_0),
			.wraddress			({ram_wr_addr[10], ram_wr_addr[6:0]}),
			.data				(ram_wr_data[31:0]),

			.rdclock			(system_clk),
			.rdaddress			({ram_rd_addr[10], ram_rd_addr[6:0]}),
			.q					(ram_b7_rd_data_0[31:0])
);

b7_ram				b7_ram_g0
(
			.wrclock			(dclk),
			.wren				(ram_b7_wr_en_0),
			.wraddress			({ram_wr_addr[10], ram_wr_addr[6:0]}),
			.data				(ram_wr_data[63:32]),

			.rdclock			(system_clk),
			.rdaddress			({ram_rd_addr[10], ram_rd_addr[6:0]}),
			.q					(ram_b7_rd_data_0[63:32])
);

b7_ram				b7_ram_b0
(
			.wrclock			(dclk),
			.wren				(ram_b7_wr_en_0),
			.wraddress			({ram_wr_addr[10], ram_wr_addr[6:0]}),
			.data				(ram_wr_data[95:64]),

			.rdclock			(system_clk),
			.rdaddress			({ram_rd_addr[10], ram_rd_addr[6:0]}),
			.q					(ram_b7_rd_data_0[95:64])
);

///			1 组 b0b6

b0b6_ram			b0b6_ram_r1
(
			.wrclock			(dclk),
			.wren				(ram_b0b6_wr_en_1),
			.wraddress			(ram_wr_addr),
			.data				(ram_wr_data[31:0]),

			.rdclock			(system_clk),
			.rdaddress			(ram_rd_addr),
			.q					(ram_b0b6_rd_data_1[31:0])
);

b0b6_ram			b0b6_ram_g1
(
			.wrclock			(dclk),
			.wren				(ram_b0b6_wr_en_1),
			.wraddress			(ram_wr_addr),
			.data				(ram_wr_data[63:32]),

			.rdclock			(system_clk),
			.rdaddress			(ram_rd_addr),
			.q					(ram_b0b6_rd_data_1[63:32])
);

b0b6_ram			b0b6_ram_b1
(
			.wrclock			(dclk),
			.wren				(ram_b0b6_wr_en_1),
			.wraddress			(ram_wr_addr),
			.data				(ram_wr_data[95:64]),

			.rdclock			(system_clk),
			.rdaddress			(ram_rd_addr),
			.q					(ram_b0b6_rd_data_1[95:64])
);

///			1 组 b7

b7_ram				b7_ram_r1
(
			.wrclock			(dclk),
			.wren				(ram_b7_wr_en_1),
		.wraddress		({ram_wr_addr[10], ram_wr_addr[6:0]}),
		.data			(ram_wr_data[31:0]),

		.rdclock		(system_clk),
		.rdaddress		({ram_rd_addr[10], ram_rd_addr[6:0]}),
		.q				(ram_b7_rd_data_1[31:0])
);

b7_ram			b7_ram_g1
(
		.wrclock		(dclk),
		.wren			(ram_b7_wr_en_1),
		.wraddress		({ram_wr_addr[10], ram_wr_addr[6:0]}),
		.data			(ram_wr_data[63:32]),

		.rdclock		(system_clk),
		.rdaddress		({ram_rd_addr[10], ram_rd_addr[6:0]}),
		.q				(ram_b7_rd_data_1[63:32])
);

b7_ram			b7_ram_b1
(
		.wrclock		(dclk),
		.wren			(ram_b7_wr_en_1),
		.wraddress		({ram_wr_addr[10], ram_wr_addr[6:0]}),
		.data			(ram_wr_data[95:64]),

		.rdclock		(system_clk),
		.rdaddress		({ram_rd_addr[10], ram_rd_addr[6:0]}),
		.q				(ram_b7_rd_data_1[95:64])
);

/*

Endmodule
```

References

[1] R. Friedel and P. Israel, *Edison's Electric Light: Biography of an Invention*, New Jersey, Rutgers University Press, 1986.

[2] T. Hughes, *American Genesis*: *A Century of Invention and Technological Enthusiasm*. Chicago, The University of Chicago Press, 1990.

[3] F. Furfari, "A different kind of chemistry: a history of tungsten halogen lamps," *Industry Applications Magazine, IEEE*, vol. 7, pp. 10-17, 2001.

[4] J.R. Brodrick, "U.S. lighting market characterization volume 1: National lighting inventory and energy consumption estimate," *U.S. Department of Energy*, September 2002.

[5] W. Elenbaas, *Fluorescent Lamps.* London, U.K.: Macmillan, 1971.

[6] J. A. Sigler, J. F. Crutchfield, and H. C. Wilins, *The Fluorescent Lamp.* New York, McGraw-hill Book Company, 1975.

[7] K. H. Butler, *Fluorescent Lamp Phosphors: Technology and Theory.* New Jersey, Pennsylvania State University Press, 1980.

[8] J. M. Phillips, M. E. Coltrin, M. H. Crawford, A. J. Fisher, M. R. Krames, R. Mueller-Mach, G. O. Mueller, Y. Ohno, L. E. S. Rohwer, J. A. Simmons, and J. Y. Tsao, "Research challenges to ultra-efficient inorganic solid-state lighting," *Laser and Photonics Reviews*, vol. 1, no. 4, pp. 307–333, November 2007.

[9] P. Flesch, *Light and Light Sources: High-Intensity Discharge Lamps*. Berlin, Germany: Springer, 2006.

[10] H. J. Round, "A note on carborundum," *Electrical World*, vol. 49, pp. 309, February 1907.

[11] C. P. Kuo, R. M. Fletcher, T. D. Osentowski, M. C. Lardizabal, M. G. Craford, and V. M. Robbins, "High performance AlGaInP visible light emitting diodes," *Applied Physics Letters*, vol. 57, no. 27, pp. 2937 –2939, Dec. 1990.

[12] S. Nakamura, T. Mukai, and M. Senoh, “Candela-class high-brightness InGaN/AlGaN double-heterostructure blue-light-emitting-diodes,” *Applied Physics Letters*, vol. 64, no. 13, pp. 1687–1689, March 1994.

[13] S. Nakamura, T. Mukai, and M. Senoh, “High-brightness InGaN/AlGaN double heterostructure blue-green-light-emitting diodes,” *Journal of Applied Physics*, vol. 76, no. 12, pp. 8189–8191, December 1994.

[14] S. Nakamura, “InGaN/AlGaN blue-light-emitting diodes,” *Journal of Vacuum Science and Technology A*, vol. 13, no. 3, pp. 705–710, May/ June 1995.

[15] Wikipedia, “Compact fluorescent lamp,” http://en.wikipedia.org/wiki, [Online; accessed 20-July-2014].

[16] C. J. Humphreys, “Solid-state lighting,” *MRS Bulletin (Special Issue on Harnessing Materials for Energy)*, vol. 33, pp. 459–470, April 2008.

[17] K. Lim, J. C. Lee, G. Panotopulos and R. Helbing, “Illumination and color management in solid state lighting” *IEEE Industry Applications Conference, 41th IAS Annual Meeting*, Tampa, Florida, USA, pp. 2616-2620, 8-12 Oct. 2006.

[18] M. S. Shur and A. Zukauskas, “Solid-state lighting: toward superior illumination,” *Proceedings of IEEE*, vol. 93, no. 10, pp. 1691–1703, Oct.2005.

[19] G. Harbers, S. J. Bierhuizen, and M. R. Krames, “Performance of high powerlight emitting diodes in display illumination applications,” *Journal of Display Technology*, vol. 3, no. 2, pp. 98–109, June 2007.

[20] Google, “LED applications,” https://www.google.com.hk/imghp?-hl=zh-TW&tab=wi, [Online; accessed 12-July-2014].

[21] S. C. Shin, Y. Jung, T. J. Ahn, S. S. Jeong, S. G. Lee, and K. Y. Choi, “The Compact Systems Design Based on DMD and the Straight Line 2-Channel LED for a Mobile Embedded Pico Projector,” *Display Technology, Journal of*, vol. 8, no. 4, pp. 219 –224, Apr. 2012.

[22] F. Fournier and J. Rolland, “Design methodology for high brightness projectors,” *Journal of Display Technology*, vol. 4, no. 1, pp. 86–91, March

2008.

[23] C. C. Chen, C. Y. Wu, and T. F. Wu, “LED back-light driving system for LCD panels,” in *Proceedings of IEEE Applied Power Electronics Conference and Exposition (APEC)*, Mar. 2006, pp. 381–385.

[24] C. C. Chen, C. Y. Wu, Y. M. Chen, and T. F. Wu, “Sequential color LED backlight driving system for LCD panels,” *IEEE Transactions on Power Electronics*, vol. 22, no. 3, pp. 919–925, May 2007.

[25] D. Gacio, J. Cardesin, E. L. Corominas, J. M. Alonso, M. Dalla-Costa, and A. J. Calleja, “Comparison among power LEDs for automotive lighting applications,” in *Record of IEEE Industry Applications Conference*, October 2008, pp. 1–5.

[26] CNET, “LED LCD backlights explain,” http://www.cnet.com/news/ led-lcd-backlights-explained, [Online; accessed 17-May-2014].

[27]S. K. Ng, K. H. Loo, Y. M. Lai, Chi K. Tse, “Color Control System for RGB LED with Application to Light Sources Suffering from Prolonged Ageing,” *IEEE Transactions on Industrial Electronics*, Vol. 61, No. 4, pp. 1788–1798, 2014

[28] V.S. Abramov, A.E. Puisha, I.P. Polyakova, M.G. Tomilin, A.I. Chuvashov “LED modules for large screens,” *Journal of Optical Technology*, vol. 70, no. 7, pp. 492–494, July 2003.

[29] F. Nguyen, “Challenges in the design of a RGB LED display for indoor applications,” *Synthetic Metals*, vol. 122, no. 1, pp. 215–219, May 2001.

[30] Li Hai Quan. Test and evaluation for CRT display. *Beijing: Posts & Telecom Press*, 1998.

[31] E.B. Bellers, I.E.J. Heynderickx, G.de Haan, I.de Weerd, “Optimal television scanning format for CRT-displays,” *IEEE Transactions on Consumer Electronics*, vol. 47, no. 3, pp. 347–353, July 2001.

[32] A. Rogalski, “Recent progress in infrared detector technologies,” *Infrared Phys. Technol.*, vol. 54, no. 3, pp. 136–154, 2011.

[33] CHEN Zhisheng, CHEN Jingxian. “A Multi-resolution VGA Display

Controller Design Based on FPGA," *Modern Electronics Technique,* vol. 13, pp.187-189, 2008.

[34] FAN Jia-li, ZHAO Guo-liang. "A design of VGA interface for the embedded video system," *Applied Science and Technology*, vol. 35, no. 10, pp. 37–40, 2008.

[35] Istvan Gorog, "Displays for HDTV: Direct-view CRT's and projection systems," *Proc. of IEEE*, vo1.82, no.4, pp.520-536, Apr. 1994

[36] C. Chuang, J. Tsai, R. Hung, P. Huang, J. Wen, "A Non-impact Full Screen CRT Purity Test Station for Monitor and ITC Operation," *SID, Display Manufacturing Technology Conference*, 1997

[37] Qinglei Lu etc. "CRT Graphic Display System of Sigma fire alarm system," *Atomic Energy Science and Technology*, 2000-3.

[38] Chang, S.C., "The TFT-LCD industry in Taiwan: competitive advantage sand future developments," *Technology in Society,* vol. 27(2), pp. 199-215, 2005.

[39] Teo, N. , R. Chu, , D. H. Lee, , M. Bang, and K. Suratkal , "LCD TVs: The next big thing," *IG Financial Markets* , November, pp. 1-25, 2003

[40] Hung, S.W., "Competitive strategies for Taiwan's thin film transistor-liquid crystal display (TFT-LCD) industry," *Technology in Social*, vol. 28, pp. 349-361, 2006.

[41] M. Schu, G. Schemer, C. Tuschen, and A. Stolze, "System on silicon-IC for motion compensated scan rate conversion, picture-in-picture processing, split screen applications and display processing," *IEEE Transactions on Consumer Electronics*, vol. 45, August 1999, pp. 842-850.

[42]F. Gatti, A. Acquaviva, L. Benini, and B. Ricco, "Low power control techniques for TFT LCD displays," *in Proceedings of the 2002 international conference on Compilers, architecture, and synthesis for embedded systems,* 2002, pp. 218-224.

[43] Zhang Da, and Xu Shu Yan, "High speed CCD image data fiber transmission system," *Optics and Precision Engineering*, vo1.17, No.3, 2009, pp.669-675.

[44]K. Park, W. Oh, B.Y. Choi, J.W. Han, and S. Park, "A 4-channel 12.5 Gb/s common-gate transimpedance amplifier array for DVI/HDMI applications," *Circuits and Systems, IEEE International Symposium on*, 2007, pp. 2192-2195.

[45] S. M. Liu, C. F. Chen, and K. C. Chou, "The design and implementation of a low-cost 360-degree color LED display system," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 2, pp. 289–296, May, 2011.

[46] Yong-Jun Kim, Byung-Joo Hong, Jun-Dong Cho, "Implementation of a LED Display System for High Quality Video Processing," *International SoC Design Conference (ISOCC)*, pp.552-555, 2009.

[47] F. Chang, X.F. Zheng, R.G. Wang, and T.F. Ding, "Dynamic nonlinear transformation algorithm in LED display panel," *in Proc. 2nd Int. Conference CINC*, 2010, pp. 39–42

[48] Svilainis L. "LED brightness control for video display application," *Displays*. 29, 2008; 506–511.

[49] Svilainis L., Dumbrava V. "LED Forward Current Errors Induced in Video Display Tile's PCB," *IEEE proceedings International Conference on Information Technology* Interfaces, Cavtat, Croatia, 2009; 565-570.

[50] Kurdthongmee W. "Design and implementation of an FPGA-based multiple colour LED display board," *Microprocessors and Microsystems*. 29, 2005; 327–336.

[51] Nguyen F. "Challenges in the design of a RGB LED display for indoor applications," *Synthetic Metals*, 122, 2001; 215–219.

[52] K. H. Pang, C. H. Chan, and T. O. Kwan, "Tricolor Light-Emitting Diode Dot Matrix Display System with Audio Output," *IEEE Transactions on Industry Applications*, vol. 37, no. 2, pp. 534-540, March, 2001.

[53]X. Lv, K. H. Loo, Y. M. Lai, Chi K. Tse, "Energy-Saving Driver Design for Full-Color Large-Area LED Display Panel Systems," *IEEE Transactions on Industrial Electronics*, Vol. 61, No. 9, pp. 4665–4673, 2014

[54] I. H. Oh, “A single-stage power converter for a large screen LCD back-lighting,” in *Proceedings of IEEE Applied Power Electronics Conference and Exposition,* March 2006, pp. 1058–1063.

[55] K. H. Loo, Y. M. Lai, S. C. Tan, and C. K. Tse, “On the color stability of phosphor-converted white LEDs under dc, PWM, and bilevel drive,” *Power Electronics, IEEE Transactions on*, vol. 27, no. 2, pp. 974–984, Feb. 2012

[56] K. H. Loo, W. K. Lun, S. C. Tan, Y. Lai, and C. Tse, “On Driving Techniques for LEDs: Toward a Generalized Methodology,” *Power Electronics, IEEE Transactions on,* vol. 24, no. 12, pp. 2967 –2976, Dec. 2009.

[57] S. C. Tan, “General *n*-Level Driving Approach for Improving Electrical-to- Optical Energy-Conversion Efficiency of Fast-Response Saturable Lighting Devices,” *Industrial Electronics, IEEE Transactions on,* vol. 57, no. 4, pp. 1342 –1353, Apr. 2010.

[58] G. H. G. Archenhold and K. Anderson, “Illumination control system,” *US Patent, No. 6,963,175 B2*, November 2005.

[59] B. Ackermann, V. Schulz, C. Martiny, A. Hilgers, and X. Zhu, “Control of LEDs,” *in Record of IEEE Industry Applications Conference*, October 2006, pp. 2608–2615.

[60] M. Dyble, N. Narendran, A. Bierman, and T. Klein, “Impact of dimming white LEDs: chromaticity shifts due to different dimming methods,” *in Proceedings of SPIE, International Conference on Solid State Lighting*, September 2005, pp. 167–171.

[61] B. Ackermann, V. Schulz, C. Martiny, A. Hilgers, and X. Zhu, “Control of LEDs,” *in Record of IEEE Industry Applications Conference*, October 2006, pp. 2608–2615.

[62] P. Narra and D. S. Zinger, “An effective LED dimming approach,” *in Record of IEEE Industry Applications Conference*, October 2004, pp. 1671–1676.

[63] W. Kurdthongmee, “Design and implementation of an FPGA-based multiple-color LED display board”, *J. Microprocess. Microsyst.*, vol. 29,no. 7, pp. 327–336, May 2005

[64] Howell W. D. “An overview of the electronic drive techniques for intensity control and colour mixing of low voltage light sources such as LEDs and LEPs,” *Application note 011*. Artistic Licence Ltd, London. 2002.

[65] M. Xichao, Z. Yuanyue, “Consecutive PWM driving video LED display system,” *in: Proceedings of ISCAS ’97*, 1997, pp. 1437–1439.

[66] W.D. Howell, “An overview of the electronic drive techniques for intensity control and colour mixing of low voltage light sources such as LEDs and LEPs,” *Application Note 011*, Artistic Licence Ltd., Berlin, 2002.

[67] L. Svilainis, “Considerations of the driving electronics of led video display,” in *Record of 29th International Information Technology Interfaces*, June 2007, pp. 431–436.

[68] L. Wai-keung, “On driving techniques for high-brightness light-emitting diodes,” Master’s thesis, The Hong Kong Polytechnic University, Nov. 2009.

[69] Pihos P., “Full-color video LED billboards feel the impact of alternative technologies,” *LEDs magazine*. 8, 2006; 29-31.

[70] Y. R. Song, C. S. Won, H. Ahn, D. Y. Han, Y. S. Choi and S. J. Lym, “The Study On Optimal Design and Optical Properties of LED Module for Full-color Displays,” *Proceedings of the 5th International Conference on Properties Applications of Dielectric Materials*, pp. 956-959, May 1997.